UNIVERSIDAD CARLOS III DE MADRID

Mecanismos para la gestión de información bibliográfica

Proyecto Fin de Carrera de Ingeniería Informática

Septiembre de 2009
Alumna: Mª Teresa Muñoz-Reja Herrero
Tutores: Ignacio Aedo Cuevas y José Carlos García Zorita

Agradecimientos

A mi tutores, por su confianza depositada en mí.

A mis padres, por su continuo apoyo durante todos estos años que me ha ayudado a no rendirme y seguir hacia delante a pesar del esfuerzo.

A mis compañeros del laboratorio, que siempre han estado dispuestos a echarme una mano durante el desarrollo de este proyecto, especialmente a Nacho, por su infinita paciencia y a David, que desde el otro lado del charco me ha asesorado con la memoria.

A Olivera, quien mejor ha entendido mi sufrimiento y esfuerzo durante estos años y que tanto apoyo y ayuda me ha prestado durante los mismos y, sobre todo, por su comprensión a lo largo de estos meses que he estado dedicada en cuerpo y alma a la elaboración de este proyecto.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

1 Introducción

Este Proyecto Fin de Carrera de la titulación de Ingeniería Informática tiene como finalidad la elaboración de una aplicación capaz de gestionar una base de datos de información catalogada, cuya documentación se recoge en esta memoria. Este capítulo introductorio tiene como propósito poner al lector en situación, por lo que se ofrece una breve explicación del problema que se ha planteado. Se explican también los objetivos que la aplicación debe cumplir y que son los que desde el LEMI (Laboratorio de Estudios Métricos de la Información), se han impuesto. Además, esta sección recoge la metodología de trabajo seguida y cómo se estructura el presente documento.

1.1 Planteamiento del problema

El problema surge desde el LEMI tras considerar que el proceso de actualización de los registros almacenados en su base de datos, requiere un esfuerzo y conocimiento que complica bastante su ejecución, lo que conlleva que los registros almacenados en la base de datos no se actualicen con tanta frecuencia como sería deseable. Para el LEMI es de vital importancia que estos registros estén siempre actualizados, ya que son utilizados para realizar distintos estudios y análisis métricos que forman parte de sus líneas de investigación. Además, necesitan cuantificar la producción de publicaciones investigativas en nuestro país, por lo que resulta interesante saber cuántos registros nuevos se añaden en cada proceso de actualización para así poder estimar las publicaciones nuevas. Por otro lado, el proceso actual se desarrolla mediante sentencias y expresiones complejas en lenguaje Perl, lo que requiere de personal cualificado para desempeñar dicho proceso, recayendo esta responsabilidad en una única persona.

1.2 Objetivos

La principal finalidad de este proyecto es desarrollar una aplicación que permita actualizar la base de datos del grupo LEMI, de manera rápida y sencilla, para que este proceso pueda ser llevado a cabo con frecuencia y poder manejar unos datos precisos. Los objetivos secundarios que debe satisfacer esta aplicación se enumeran a continuación:

- **Simplificar el proceso de inserción y actualización de los datos:** en el campo de la investigación se publican continuamente nuevos artículos, por lo que se desea tener actualizada esta información. Además, cierta información bibliográfica como el número de veces que un artículo es citado en otras publicaciones varía continuamente y es deseable tenerlo actualizado para que los estudios que se realicen sean precisos. Por estos motivos es importante que el proceso de inserción y actualización sea sencillo, para poder realizarlo con frecuencia y mantener los datos actualizados.
- **Proporcionar información del proceso de actualización:** no sólo interesa que la aplicación permita ejecutar el proceso de actualización de manera sencilla, sino también obtener información de cómo se ha llevado a cabo dicho proceso, para saber el número de artículos nuevos que se han añadido a la base de datos y cuáles han sido simplemente modificados. Esto es importante para evaluar la producción científica en el tiempo.
- **Facilitar las consultas a la base de datos:** la aplicación ha de ser manejada por personal que no tiene porqué tener conocimientos amplios sobre el uso de la base de datos, pero que necesitan consultar los datos almacenados en la misma, por ello el proceso de consulta deberá ser lo más intuitivo y sencillo posible.

- **Multiplataforma:** la aplicación debe poder se ejecutada en cualquier plataforma, especialmente en los sistemas operativos Windows y Mac, que son los usados por el LEMI, por ello se podrá acceder a la misma a través de distintos navegadores independientes de la plataforma.

1.3 Metodología

Con el fin de desarrollar un buen Proyecto Fin de Carrera, es necesario establecer una metodología de trabajo y ajustarse a ella. Es importante reseñar que la metodología a seguir se diferencia de cualquiera que se pueda aplicar a un proyecto software, ya que en estos últimos no existe la figura del tutor y no existe una fase de planteamiento del problema. La metodología seguida se divide en cuatro fases diferenciadas, las cuales se describen a continuación:

- **Planteamiento del problema:** esta fase abarca desde que el LEMI explica el problema que tiene hasta que se esboza una solución viable. En esta fase se producen reuniones con el representante del LEMI y con el tutor, con el fin de centrar el problema todo lo posible y analizar las posibles soluciones que se plantean. De entre todas estas soluciones, se elige la más viable y que satisfaga los criterios del cliente.
- **Desarrollo de la solución:** en esta fase se implementará la aplicación. Una vez decididas las tecnologías que son más adecuadas para el desarrollo del proyecto, éstas se utilizarán para la implementación del mismo. Esta es quizá, la fase más complicada del proyecto, ya que conlleva bastante tiempo el desarrollar toda la aplicación conforme a las necesidades del cliente y con una calidad admisible. Además, requiere de un proceso de adaptación y aprendizaje de las tecnologías, que suele ser difícil de estimar de antemano.
- **Evaluación de la solución:** en esta fase se establecen y se ejecutan las pruebas sobre la aplicación para constatar que cumple con la funcionalidad requerida. De esta forma se puede asegurar la validez de la solución desarrollada. También se requerirá de la participación del cliente, ya que su opinión es en última instancia la más importante.
- **Desarrollo de la documentación:** esta fase recoge el proceso de elaboración de la presente memoria, recopilando las cuestiones más importantes referentes al proyecto. La memoria se desarrollará a la vez que el resto de fases identificadas, ya que no es necesario esperar a haber evaluado la aplicación, para poder escribir parte de la memoria. De esta forma se redacta la memoria teniendo más frescos los conceptos relativos al desarrollo de la aplicación y se hace más llevadero todo el proceso.

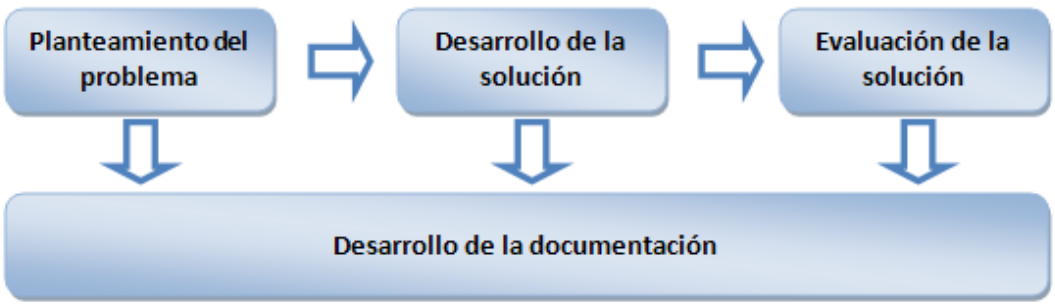

Figura 1: Metodología de trabajo

El definir y seguir una metodología no es suficiente para llevar el proyecto a buen puerto, también es necesario establecer una planificación realista donde se sitúen las distintas fases a desarrollar. Esta planificación se define en el momento del planteamiento del problema, teniendo presente cuál va a ser la fecha aproximada de entrega del proyecto. Aunque es muy difícil cumplir estrictamente los plazos marcados, ya que durante el desarrollo del proyecto surgen complicaciones imprevistas, mientras que otras tareas para las se calculaba más tiempo resultan más sencillas de lo esperado, se ha de intentar seguir la planificación de la manera más fiel posible. Al final de este documento se recogen tanto la planificación inicial como la real, para poder comparar la diferencia existente entre la planificación real y la estimada inicialmente y analizar qué tareas han llevado más tiempo del previsto y cuáles menos.

1.4 Estructura del trabajo

Este trabajo se ha estructurado en torno a seis capítulos principales, que se acompañan a su vez de una sección de bibliografía y de varios anexos que aparecen al final del presente documento. A continuación se indican los capítulos de los que se compone este proyecto y se ofrece una breve descripción de los mismos con el fin de orientar al lector en la lectura del presente documento:

- **Introducción:** en este capítulo se presenta el proyecto, explicando el problema que ha dado lugar al trabajo, los objetivos que pretende satisfacer y la metodología seguida en el desarrollo del proyecto.
- **Estudio del problema:** este capítulo aborda el contexto en el que surge el problema, hace un estudio de las distintas tecnologías existentes para hacerle frente y finalmente plantea de manera formal el problema que se ha de resolver mediante el presente proyecto, así como las limitaciones a las que habrá que hacer frente.
- **Gestión del proyecto:** este capítulo se dedica al proceso de planteamiento, ejecución y control del proyecto para alcanzar un correcto equilibrio en cuanto a coste, plazos y calidad. Además se hace un análisis de los posibles riesgos que pudieran surgir durante el desarrollo del proyecto y cómo hacerles frente. Finalmente, se establece un plan de pruebas para asegurar que el sistema cuenta con la calidad requerida y que se satisfacen todas las necesidades del cliente.
- **Solución:** aquí se describe la solución adoptada para resolver el problema, el capítulo se centra en el proceso de desarrollo seguido para alcanzar la situación, describiendo cada uno de los pasos seguidos y las decisiones tomadas.
- **Evaluación:** en este capítulo se demuestra la validez de la solución planteada, a través de las pruebas realizadas para comprobar el correcto funcionamiento, así como la calidad de la aplicación.
- **Conclusión:** en este capítulo se recogen las conclusiones obtenidas tras el desarrollo completo de este proyecto. Se sentarán además las bases para el desarrollo de trabajos futuros y se hará una revisión de los problemas encontrados en el desarrollo del proyecto.

Además, se han incluido una serie de anexos que recogen las distintas versiones que se han efectuado de la memoria, el seguimiento que se ha hecho del proyecto (incluyendo tanto la planificación inicial como la real), el manual de usuario que acompaña a la aplicación y el prototipo que se presentó al usuario para definir el diseño de la interfaz.

Estudio del problema

2 Estudio del problema

En este capítulo se presenta el contexto del problema que este proyecto trata de solucionar, para pasar posteriormente a hacer una revisión de las posibles tecnologías a utilizar en la solución del problema y finalizar definiendo las limitaciones a las que tendremos que hacer frente.

2.1 El contexto del problema

El departamento de Biblioteconomía de la Universidad Carlos III de Madrid tiene como misión investigar sobre todo tipo de tareas relacionadas con la gestión de la información en cualquier ámbito: cómo seleccionar, buscar, encontrar y organizar la información existente en instituciones tradicionales como archivos, bibliotecas y centros de documentación; pero también en otro tipo de soportes tradicionales y redes informáticas más a la orden del día, como Internet. Una de las líneas de investigación a la que dedica grandes esfuerzos es a la medición de la investigación tanto dentro como fuera de España. Enmarcado dentro del departamento de Biblioteconomía se sitúa el Laboratorio de Estudios Métricos de Información (LEMI) que recoge entre sus objetivos la investigación de varios campos relacionados con los estudios cuantitativos de la información, con el fin de avanzar adquiriendo conocimientos tanto teóricos como prácticos dentro de esos campos. Su misión principal es el estudio y evaluación de la producción y el consumo de información en ciencia y tecnología por parte de comunidades científicas, así como el análisis de las características y el comportamiento de grupo de usuarios en su interacción con los recursos de información a su disposición.

Dentro de este contexto, el LEMI hace uso de una plataforma de la empresa Thomson Scientific basada en tecnología web, la Web of Knowledge (WoK), que está formada por una amplia colección de bases de datos bibliográficas, citas y referencias de publicaciones científicas de cualquier disciplina del conocimiento, tanto científico, como tecnológico, humanístico y sociológico, desde 1945 [32]. El WoK integra en sus principales bases de datos Web of Science y Current Contents Connect, fuentes adicionales de contenido con recursos Web, con otros datos académicos y material de publicaciones, así como congresos, patentes y actas y herramientas de evaluación del rendimiento. Los productos contratados por el FECYT (Fundación Española para la Ciencia Y la Tecnología) del WoK son los siguientes:

- Productos de Citas y Actualización
  - Web of Science, desde 1900
  - Current Contents Connect
- Productos Especializados
  - ISI Proceedings (Actas, conferencias)
  - Derwent Innovations Index (patentes)
  - Medline (literatura médica)
- Productos para Análisis y Evaluación
  - Journal Citation Reports
  - Essential Science Indicators
- Gestor de Referencias Bibliográficas
  - EndNote Web

Parte del trabajo del LEMI es recoger y clasificar información obtenida de dos fuentes principales del WoK, la base de datos Web of Science y el Journal Citations Report:

- La Web of Science es una base de datos de referencias bibliográficas de artículos de revistas, que ofrece el acceso a información actual y retrospectiva de resúmenes de autor e índices de citas de cerca de 9.300 publicaciones internacionales en los campos de las ciencias, ciencias sociales, artes y humanidades.
- El Journal Citations Report (JCR) presenta datos estadísticos de citas desde 1997 en adelante, proporcionando una manera sistemática y objetiva de determinar la importancia relativa de las revistas dentro de sus categorías temáticas (factor de impacto de las revistas). Se presenta en edición de ciencias y también de ciencias sociales y ayuda a medir la influencia de la investigación mostrando las relaciones entre las revistas citadas y las que se citan.

La información de estas bases de datos se actualiza cada cierto tiempo, por lo que el LEMI debe recopilarla puntualmente y añadirla a su propia base de datos para mantenerla actualizada.

2.2 El estado de la cuestión

El objetivo de esta sección es poner al lector en el contexto de realización del trabajo. Se presenta una revisión de las tecnologías y herramientas utilizadas en el desarrollo de este proyecto: herramientas de desarrollo de aplicaciones web y sistemas gestores de bases de datos. A lo largo del capítulo no sólo se presentarán diversas tecnologías y herramientas sino que se hará una comparativa entre ellas.

Los gestores de bases de datos son una parte fundamental en el estudio y evaluación de la producción y consumo de información por parte de las comunidades científicas. Gracias a ellos la información se puede almacenar de forma que la manipulación de la misma para su posterior medición y análisis resulte eficiente y sencilla. Dentro de este contexto las aplicaciones web desempeñan un papel fundamental para ayudar a los usuarios a trabajar con las bases de datos que almacenan información documental sin necesidad de tener conocimientos específicos sobre bases de datos. Esta sección hace una revisión sobre los gestores de bases de datos, dando una visión general sobre su uso y características, para posteriormente hacer una comparativa entre los más utilizados actualmente. El objetivo es comprender mejor las ventajas que presentan los gestores de bases datos en el almacenamiento y gestión de información documental de carácter académico. Además se analizarán en profundidad las aplicaciones web debido a la importancia que cobran en el ámbito del manejo de la información almacenada en bases de datos documentales.

2.2.1 Sistemas de Gestión de Bases de Datos

Un sistema gestor de bases de datos (SGBD) es un conjunto de programas de ordenador que controla la creación, mantenimiento y uso de la base de datos de una organización y sus usuarios finales [26]. Un SGBD debe cumplir los siguientes objetivos [26]:

- **Abstracción de la información:** para el usuario debe ser transparente la manera física en que están almacenados los datos, ya que no tiene que conocer este tipo de detalles. Para ellos los SGBD proporcionan varios niveles de abstracción.
- **Independencia:** se trata de la capacidad que ofrece un SGBD de modificar el esquema de una base de datos sin tener que realizar cambios en todas aquellas aplicaciones que se sirvan de ella.
- **Consistencia:** cuando no se consigue eliminar la redundancia, es necesario controlar que la información que se almacena de manera repetida se actualice de

manera simultánea. Los SGBD proporcionan herramientas que permiten controlar este tipo de situaciones.

- **Seguridad:** se debe garantizar que la información almacenada esté segura, para ello los SGBD deben proporcionar herramientas de gestión y concesión de permisos de distintas categorías.
- **Manejo de transacciones:** se entiende por transacción un programa que se ejecuta como una sola operación. Sin embargo, debido a distintos factores puede que una transacción no llegue a buen término. En ese caso los SGBD deben asegurarse de que la base de datos queda en modo consistente.
- **Tiempo de respuesta:** es deseable minimizar el tiempo que el SGBD tarda en proporcionar la información solicita y en almacenar los cambios realizados. Esto es especialmente importante en aplicaciones web donde el cliente espera una respuesta inmediata por parte de la aplicación.

Los sistemas gestores de bases de datos cuentan con numerosas ventajas, todas ellas relacionadas con facilitar la manipulación de grandes volúmenes de datos:

- **Aumenta la integridad de los datos:** la integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.
- **Mejora la seguridad:** la seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros. Sin embargo, los SGBD permiten mantener la seguridad mediante el establecimiento de claves para identificar al personal autorizado a utilizar la base de datos. Las autorizaciones se pueden realizar a nivel de operaciones, de modo que un usuario puede estar autorizado a consultar ciertos datos pero no a actualizarlos, por ejemplo.
- **Incrementa la accesibilidad:** muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.
- **Aumento de la productividad:** el SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación. El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel. Muchos SGBD también proporcionan un entorno de cuarta generación consistente en un conjunto de herramientas que simplifican, en gran medida, el desarrollo de las aplicaciones que acceden a la base de datos. Gracias a estas herramientas, el programador puede ofrecer una mayor productividad en un tiempo menor.
- **Facilita el mantenimiento gracias a la independencia de datos:** en los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan. Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados. Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos,

gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

- **Aumento de la concurrencia:** en algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o, incluso, que se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.
- **Mejora en los servicios de copias de seguridad y de recuperación ante fallos**: muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones. Los usuarios tienen que hacer copias de seguridad cada día, y si se produce algún fallo, utilizar estas copias para restaurarlos. En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

Sin embargo, los SGBD también presentan determinados inconvenientes que deben ser tenidos en cuenta a la hora de decantarnos por su uso. Uno de los principales es que es necesario disponer de una o más personas que administren la base de datos, lo que puede incrementar el coste del sistema final. Además, debido a la complejidad y el tamaño que ocupan los SGBD es necesario disponer de una cantidad de memoria suficiente para que puedan correr sin problemas.

Los distintos SGBD se clasifican en función de la representación lógica de los datos que muestran a los usuarios. Estos modelos han ido evolucionando y cambiando a lo largo de los años, siendo en la actualidad los más utilizados el modelo relacional y el orientado a objetos. El modelo relacional se basa en la teoría relacional de las matemáticas, siendo la estructura principal las tablas bidimensionales de filas y columnas. Cada línea (tupla) representa una entidad a almacenar en la base de datos. Las características de cada entidad están definidas por las columnas de las relaciones (atributos). Aquellas entidades con características comunes, es decir, definidas por el mismo conjunto de atributos, formarán parte de la misma relación. El modelo orientado a objetos está representado por un conjunto de clases que definen las características y el comportamiento de los objetos que poblarán la base de datos. A diferencia de las bases de datos tradicionales, las orientadas a objetos almacenan no sólo los datos solicitados por las aplicaciones sino también las operaciones posibles con tales datos. Este modelo es una buena elección para aquellos sistemas que necesitan un buen rendimiento en la manipulación de datos complejos.

Revisión de gestores de bases de datos

Una vez analizadas las características que definen a los gestores de bases de datos, vamos a estudiar algunos de los más utilizados hoy en día y que se ajustan al modelo relacional, concretamente MySQL, Oracle y PostgreSQL para posteriormente hacer una comparativa entre ellos. El motivo por el que nos hemos centrado en los gestores relacionales ha sido las múltiples ventajas que presenta el modelo relacional frente a otros. Entre estas ventajas destaca la garantía de no duplicidad de registros y la integridad referencial. Además favorecen la normalización por ser más comprensibles y aplicables.

MySQL

Se trata de un sistema de gestión de base de datos relacional, multihilo y multiusuario desarrollado en su mayor parte en ANSI C. MySQL es muy utilizado en aplicaciones web, en

plataformas LAMP y por herramientas de seguimiento de errores [15]. Las características principales con las que cuenta MySQL son las siguientes:

- **Transacciones que cumplen con la funcionalidad ACID:** esta característica aporta una mayor fiabilidad a las bases de datos, ya que se incrementa su seguridad al facilitar el proceso de recuperación en caso de que se produzca algún fallo en el servidor que impida ejecutar la consulta de manera completa. De esta forma la base de datos se mantiene siempre en estado consistente ya que las transacciones se ejecutan completamente o no se ejecutan.
- **Replicación:** se permite a las bases de datos de un servidor MySQL ser duplicadas en otro, de esta forma se aumenta la disponibilidad de los datos y se mejora el rendimiento de las consultas globales. La replicación en MySQL se basa en un servidor maestro que toma nota de todos los cambios en las bases de datos (actualizaciones, borrados, y así) en los logs binarios. Cada servidor esclavo recibe del maestro las actualizaciones guardadas que el maestro ha guardado en su log binario, de forma que el esclavo puede ejecutar las mismas actualizaciones en su copia de los datos.
- **Soporte de bases de datos y tablas de gran tamaño:** usando el motor de almacenamiento MyISAM, el máximo tamaño de las tablas es de 65536 terabytes. Por lo tanto, el tamaño efectivo máximo para las bases de datos en MySQL usualmente lo determinan los límites de tamaño de ficheros del sistema operativo, y no por límites internos de MySQL.
- **Búsqueda por texto completo:** este tipo de búsquedas tienen la ventaja que permiten devolver sólo los registros que satisfacen el texto de la búsqueda y no aparecen acompañados de otros términos, en vez de mostrar todos los registros donde aparezca el texto a buscar además de otros términos.
- **Subconsultas:** Una subconsulta es una consulta anidada en una instrucción SELECT, INSERT, UPDATE o DELETE, o bien en otra subconsulta. Las subconsultas se pueden utilizar en cualquier parte en la que se permita una expresión. Algunas de sus ventajas son que permiten aislar cada parte de un comando y además proporcionan un modo alternativo de realizar operaciones que de otro modo necesitarían joins y uniones complejas.

MySQL cuenta con numerosas ventajas, a continuación se definen algunas de las más importantes:

- **Rapidez en la lectura de datos:** se trata de un SGBD muy rápido en cuanto a lectura cuando utiliza el motor no transaccional MyISAM, pero esto puede provocar problemas de integridad en entornos de alta concurrencia de modificación. En el caso de las aplicaciones web, hay una baja concurrencia en la modificación de los datos y en cambio la lectura de datos es intensiva, lo que hace a MySQL muy apropiado para este tipo de aplicaciones.
- **Elevado rendimiento:** MySQL presume de su alto rendimiento y su velocidad, tanto que sólo se añaden nuevas características al mismo si estas no afectan a su rendimiento, sino estas características se mantienen a la espera hasta que se consigue encontrar el punto en que no afecten al rendimiento total del sistema.
- **Uso gratuito:** otra de las ventajas indudables de MySQL es su precio, ya que para la mayoría de los propósitos se trata de una aplicación gratuita con licencia GPL, lo que permite usar el software, modificarlo e incluso redistribuirlo. En caso de querer distribuir de manera comercial MySQL como parte de otra aplicación, es

necesario pagar una licencia comercial. Por lo tanto, MySQL se rige bajo un esquema de doble licencia dependiendo de los propósitos del usuario.

- **Alta estabilidad:** los desarrolladores de MySQL han considerado la estabilidad de vital importancia, por lo que antes de lanzar versiones nuevas las someten a una serie de controles y pruebas. Además, al ser código abierto cuentan con el respaldo de una gran comunidad de desarrolladores y clientes que notifican y reparan cualquier tipo de error tras probarlo en diversas plataformas y entornos.
- **Facilidad de uso:** MySQL es su facilidad de uso, no precisa de complicados procedimientos de configuración, por defecto viene una sencilla configuración que simplifica el proceso de instalación. Para requisitos más específicos como añadir contraseñas de seguridad, basta con seguir los pasos de una serie de pantallas.

Sin embargo, MySQL también carece de cierta funcionalidad, ya que hasta la versión 5.0 no ha incluido las opciones de vistas, procedimientos almacenados y disparadores, y aún incluyéndolos, estas tres características presentan una funcionalidad limitada y están sujetos a ciertas restricciones [24].

Oracle

Se trata de un sistema de gestión de bases de datos relacional desarrollado por Oracle Corporation. Es considerado como uno de los sistemas de bases de datos más completos, destacando por su soporte de transacciones, estabilidad, escalabilidad y su soporte multiplataforma. Es un producto vendido a nivel mundial, aunque debido a su elevado precio no está muy extendido su uso en el desarrollo de aplicaciones web [18].

Todas las versiones de Oracle incluyen lenguajes e interfaces que permiten a los programadores acceder y manipular los datos almacenados en las bases de datos. Los datos en Oracle pueden ser accedidos usando SQL [28], ODBC [17], JDBC [12], SQLJ [27], OLE DB [16], ODP.NET [19], SQL/XML [28], XQuery [34] y WebDAV [33]. Los programas desplegados dentro de la base de datos pueden ser escritos en PL/SQL y en Java. Además proporciona soporte a estructuras orientadas a objetos.

Debido a la potencia de Oracle, presenta un gran rendimiento incluso en entornos exigentes, es por ello una de las bases de datos más empleadas por las grandes empresas. Aún así, cada aplicación tiene unos requisitos diferentes de rendimiento, para lo que Oracle proporciona distintas herramientas que permiten optimizarlo según las necesidades de cada cliente. Para mejorar el rendimiento incluye mecanismos para aumentar la concurrencia, la consistencia en lectura y herramientas de bloqueo.

Presenta también herramientas que pueden ser de gran utilidad en el entorno de la inteligencia de negocio, como son los almacenes de datos, la minería de datos, OLAP, ejecución en paralelo, y SQL analítico entre otros.

Oracle integra los últimos avances en áreas de seguridad y privacidad, reduciendo los riesgos de amenazas externas. Además, proporciona un mecanismo de encriptación de datos sensibles sin la necesidad de hacer cambios sobre el código de la aplicación.

En cuanto al almacenamiento, éste puede ser gestionado de manera automática y optimizado de manera transparente. Oracle minimiza el coste de las operaciones de entrada y salida, reduce la cantidad de espacio de almacenamiento y maximiza el rendimiento.

Una de las principales desventajas de Oracle, junto a su precio, es que su uso no es intuitivo, ya que requiere una formación previa para manejarlo de manera eficiente. No basta con instalar Oracle, sino que hay que adaptarlo a las necesidades propias o las operaciones sobre el mismo se harán muy lentas.

PostgreSQL

Se trata de un sistema de gestión de bases de datos relacional, orientado a objetos y de software libre, publicado bajo la licencia BSD [23]. A continuación se describen alguna de las principales características de PostgreSQL:

- **Alta concurrencia:** una de sus características principales es su alta concurrencia gracias a un sistema denominado MVCC (Acceso concurrente multi-versión, por sus siglas en inglés). PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.
- **Amplia variedad de tipos de datos:** otra de sus ventajas es la amplia variedad de tipos nativos a los que da soporte: números de precisión arbitraria, texto de largo ilimitado, figuras geométricas, direcciones IP, bloques de direcciones estilo CIDR, direcciones MAC y arrays. Además los usuarios pueden crear sus propios tipos de datos.
- **Soporte para disparadores y vistas:** al igual que Oracle, PostgreSQL incluye la funcionalidad necesaria para definir disparadores que pueden usarse en combinación con tablas pero no con vistas. Las vistas, sin embargo, pueden definirse mediante reglas. En caso de que existan múltiples disparadores estos se ejecutan en orden alfabético. Los disparadores además de invocar funciones escritas en el lenguaje PL/PgSQL pueden invocar otras escritas en diferentes lenguajes como pueda ser PL/Perl. Estas funciones aumentan la versatilidad de la base de datos.
- **Herencia:** PostgreSQL hace uso de la herencia, permitiendo que las tablas puedan heredar sus características de una tabla “padre”. Los datos que aparecen en las tablas hijas deben aparecer en las tablas padre, a no ser que se añada la palabra clave ONLY. El hecho de añadir una columna nueva en la tabla padre hace que esa misma columna aparezca en la tabla que hereda de ella. El uso de herencia puede ser muy útil para implementar partición de tablas usando disparadores o bien reglas para insertar directamente desde la tabla padre a la tabla hija apropiada.
- **Extensibilidad:** al ser código abierto si hay alguna funcionalidad de la que PostgreSQL carece puede ser añadida por el propio programador. Además en Internet existen múltiples paquetes que añaden funcionalidad a este gestor de bases de datos. Un ejemplo de paquete que se puede encontrar y añadir a PostgreSQL es un conjunto de tipos de datos geográficos que pueden ser usados de manera eficiente en modelos espaciales tipo GIS.

Una de sus mayores desventajas es que no presenta un rendimiento tan alto como MySQL, ya que el hecho de incluir tanta funcionalidad retarda la ejecución de las operaciones. Por lo que debe ser tenido en cuenta a la hora de valorar si lo que buscamos realmente es aumentar la funcionalidad de nuestra base de datos o si nos es imprescindible mantener un buen rendimiento.

2.2.1.1 Comparativa entre MySQL, Oracle y PostgreSQL

Como hemos podido ver hasta ahora, estos tres gestores de bases de datos comparten muchas características pero también existen numerosas diferencias entre ellas, principalmente en cuanto a funcionalidad, rendimiento y precio. La Tabla 1 muestra una comparativa entre las tres para apreciar mejor las similitudes y diferencias ente MySQL, Oracle y PostgreSQL. Nos hemos fijado a la hora de establecer la comparación en los siguientes criterios:

- **Licencia:** el motivo de analizar las licencias ha sido la importancia de las mismas, puesto que determinan la normativa a la que estamos sujetos durante su uso.
- **Rendimiento:** característica fundamental a la hora de usar un SGBD en combinación con una aplicación web y no hacer esperar al usuario cuando éste solicita datos almacenados en la base de datos.
- **Estabilidad:** es un elemento de suma importancia ya que nos permite hacernos una idea de la resistencia a fallos que ofrece el SGBD.
- **Facilidad de uso:** es importante que el SGBD sea fácil de utilizar y de mantener, ya que requerirá invertir menos recursos en el aprendizaje del manejo de la misma.
- **Documentación disponible:** es importante contar con numerosa documentación a nuestro alcance para poder recurrir a ella en caso de que surja algún tipo de duda o problema durante el uso y mantenimiento del SGBD.
- **Comunidad de usuarios:** si el SGBD cuenta con una comunidad de usuarios importante, ya que ofrece información sobre el SGBD mediante tutoriales, artículos, noticias y listas de correo, que pueden ser de gran utilidad en caso de dudas.
- **Seguridad:** característica crítica ya que asegura la protección de la base de datos y la información contenida en la misma frente a usuarios no autorizados o con fines dañinos.
- **Integridad de datos:** es necesario asegurarse de que la información almacenada en la base de datos es correcta y completa y que no se ha perdido durante la ejecución de ninguna sentencia.
- **Concurrencia:** el SGBD debe asegurar cierta concurrencia para que distintos usuarios puedan realizar operaciones sobre la base de datos al mismo tiempo, esto cobra especial importancia en las aplicaciones web donde existen varios usuarios intentando acceder a la información al mismo tiempo.
- **Transacciones:** el soporte de transacciones por parte del SGBD asegura tras la ejecución de una serie de consultas la base de datos quede en modo consistente tanto si se ejecuta correctamente la transacción como si ocurre algún fallo durante su ejecución.
- **Procedimientos almacenados:** la ventaja de que los procedimientos almacenados sean soportados por el SGBD es que su ejecución en respuesta a una petición de usuario se desarrolla directamente en el motor de la base de datos, accediendo directamente a los datos que necesita manipular y aligerando el envío de datos de entrada y salida.
- **Disparadores:** su uso mejora la administración de la base de datos, ya que pueden generar valores de columnas, prevenir errores de datos, sincronizar tablas, entre otras funciones.
- **Subconsultas:** entre sus ventajas está que permiten aislar cada parte de un comando y facilitar algunas operaciones que de otra forma se volverían muy complejas.
- **Vistas:** esta característica nos permite reflejar el contenido de una o más tablas, accediendo al mismo como si de una tabla se tratara, aumentando la seguridad y la comodidad al acceder a dichos datos.

- **Replicación:** posibilidad de duplicar las bases de datos almacenadas en un servidor en otro para aumentar la disponibilidad de los datos y el rendimiento.

	MySQL	Oracle	PostgreSQL
Licencia	GPL	Privativa	BSD
Rendimiento	Alto	Medio-Alto	Medio-Alto
Estabilidad	Alta	Alta	Media
Facilidad de uso	Alta	Media	Media-Alta
Documentación disponible	Alta	Alta	Media-Baja
Comunidad de usuarios	Alta	Alta	Baja
Seguridad	Alta	Alta	Media-Alta
Integridad de datos	Media-Alta	Alta	Alta
Concurrencia	Media	Alta	Alta
Transacciones	Sí	Sí	Sí
Procedimientos almacenados	Sí pero con restricciones	Sí	Sí
Disparadores	Sí pero con restricciones	Sí	Sí
Subconsultas	Sí	Sí	Sí
Vistas	Sí pero con restricciones	Sí	Sí
Replicación	Sí	Sí	Sí

Tabla 1: Comparativa entre MySQL, Oracle y PostgreSQL

Como conclusión a los resultados mostrados en la Tabla 1 se puede considerar Oracle como el SGBD más completo, sin embargo, su elevado precio es su principal inconveniente, que debe ser tenido en cuenta a la hora de valorar si las características que aportan son realmente esenciales para la aplicación a desarrollar. PostgreSQL se alza como una de las principales alternativas en código libre y con gran funcionalidad, aunque como todos los SGBD también presenta ciertas desventajas como que su uso no está tan extendido como el de Oracle y MySQL y por lo tanto está menos documentado, además su rendimiento es inferior al desempeñado por MySQL. Si lo que se está buscando es un alto rendimiento, y ese es el caso de las aplicaciones web, en código libre, la mejor opción es MySQL, ya que presenta una elevada concurrencia y un alto rendimiento, a pesar de contar con ciertas restricciones en otras áreas.

Para aprovechar mejor las características que ofrecen los sistemas gestores de bases de datos, a menudo se usan combinados con aplicaciones web. El motivo por el que se usan las aplicaciones web en este contexto es que facilitan a los usuarios el manejo y gestión de los datos almacenados en la base de datos sin necesidad de tener unos amplios conocimientos de cómo funciona internamente la base de datos. Esta es la razón que nos lleva a analizar las características de las aplicaciones web y algunas de las herramientas existentes para facilitar su desarrollo.

2.2.2 Herramientas para el desarrollo de aplicaciones web

Se considera una aplicación web a cualquier tipo de aplicación a la que los usuarios pueden acceder mediante un navegador a través de Internet o de una intranet [2].Entre sus principales ventajas destaca su facilidad para ser actualizadas y mantenidas, sin la necesidad de tener que distribuir e instalar software a cada uno de los usuarios potenciales. Además, permiten la interacción de manera dinámica con el usuario. A continuación se va a realizar un análisis de las aplicaciones web, en concreto se analizará sus características, arquitectura y herramientas

de desarrollo, centrándonos en el estudio de distintos frameworks que facilitan la implementación de aplicaciones web.

2.2.2.1 Características de las aplicaciones web

Las aplicaciones web se caracterizan por su compatibilidad multiplataforma. El uso de tecnologías como PHP, Java, Flash, ASP y Ajax permite un desarrollo efectivo de programas, soportando todos los sistemas operativos principales. Otra de sus características principales es su inmediatez de acceso, ya que no necesitan ser descargadas, instaladas y posteriormente configuradas, sino que se encuentran disponibles para trabajar independientemente de la configuración o hardware del usuario. Además, siempre se encuentran actualizadas a la última versión sin que el usuario tenga que tomar medida alguna o interferir en sus hábitos. Por otro lado, las aplicaciones basadas en web tienen menos requerimientos de memoria RAM de parte del usuario final que los programas instalados localmente, ya que residen y se ejecutan en los servidores del proveedor en vez de en el ordenador del usuario, lo que permite al usuario que pueda estar ejecutando otros programas a la vez.

Pero no todo son ventajas en cuanto a las aplicaciones web, ya que estás cuentan también con ciertas carencias. Entre ellas cabe destacar las diferencias de presentación entre plataformas y navegadores, lo cual es el resultado de la ausencia de estándares ampliamente soportados que dificulta el desarrollo de las aplicaciones. Las aplicaciones web también presentan un acceso limitado, ya que la necesidad de conexión permanente y rápida a Internet hace que el acceso a las mismas no esté al alcance de todos. Además, la interactividad no se produce en tiempo real ya que en las aplicaciones web, cada acción del usuario conlleva un tiempo de espera, a veces excesivo, hasta que se obtiene la reacción del sistema.

2.2.2.2 Arquitectura

Según la definición de IEEE, la arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, así como los principios que orientan su diseño y evolución [3]. Normalmente, las aplicaciones web siguen el patrón de arquitectura Modelo-Vista-Controlador (MVC), que se encarga de separar la parte de datos de la aplicación, de la interfaz de usuario y la lógica de control mediante tres componentes diferentes [20]. Los componentes que forman parte de este patrón son los siguientes:

- **Modelo:** representación de la información que maneja la aplicación. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista:** representación del modelo en forma gráfica, disponible para la interacción con el usuario. En el caso de las aplicaciones web, la vista es la página HTML sobre la que el usuario puede realizar operaciones.
- **Controlador:** es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso de ser necesario.

La siguiente ilustración muestra de manera gráfica las diferentes relaciones entre los componentes de este patrón así como las responsabilidades de cada uno de ellos.

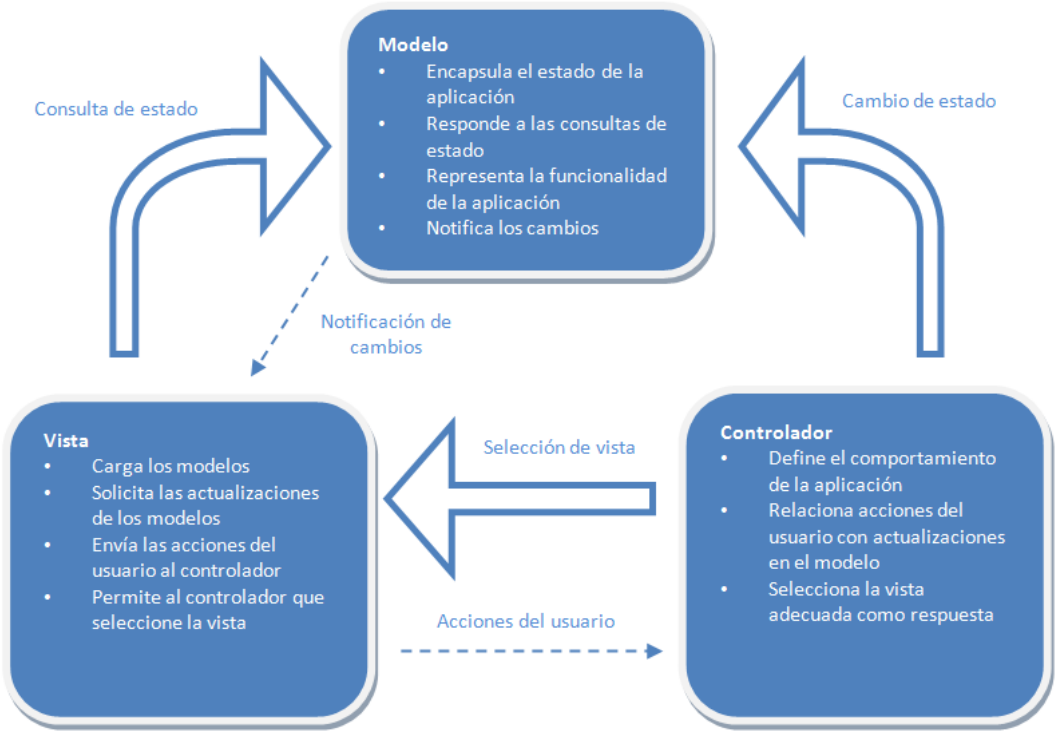

Figura 2: Relaciones y responsabilidades de los componentes del patrón MVC

Este patrón de arquitectura cuenta con numerosas ventajas, de ahí que sea tan empleado para el desarrollo de aplicaciones web y que existan multitud de frameworks que faciliten su implementación. Algunas de sus ventajas son las siguientes:

- Separación del modelo de la vista, es decir, separación de los datos de la representación visual de los mismos.
- Simplifica la inserción de múltiples representaciones de los mismos datos o información.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación debido a la independencia de capas.
- Permite la programación de las diferentes capas de manera paralela e independiente gracias a la autonomía de funcionamiento.
- Facilita el mantenimiento en caso de errores, así como las diferentes pruebas de funcionamiento del sistema.

Sin embargo, también hay que tener en cuenta que presenta ciertas dificultades a considerar:

- Desarrollar una aplicación que implemente el patrón de diseño MVC conlleva más tiempo que una aplicación que no implemente. No obstante, este tiempo se recupera en la fase de mantenimiento ya que el uso de este patrón lo simplifica.
- El patrón MVC requiere la existencia de una arquitectura inicial sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de la aplicación.
- MVC es un patrón de diseño orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.

2.2.2.3 Frameworks para el desarrollo web

A medida que las aplicaciones web aumentaron su popularidad y uso, se hizo indispensable asegurar su estabilidad, reutilización de código y la estandarización de prácticas. Debido a estas necesidades surgieron los frameworks para el desarrollo web. Un framework en términos de Software es un diseño reusable de un sistema para facilitar problemas complejos que presentan comúnmente proyectos o aplicaciones software [6].

Un framework de aplicaciones web es una potente librería software que usa como base para desarrollar aplicaciones web avanzadas 0. Normalmente contiene una arquitectura bien definida así como un conjunto de componentes reusables diseñados para simplificar el proceso de desarrollo, aumentar la consistencia y productividad y mejorar la calidad final de la aplicación. Entre las características típicas que suelen incluir están una arquitectura modular, gestión de usuarios, seguridad y roles, organización y navegación del sitio y acceso a los datos. A continuación se van a describir algunas de las principales características que acompañan a los frameworks de desarrollo web:

- **Seguridad:** muchos de los frameworks para el desarrollo de aplicaciones web vienen con gestores de autenticación y autorización que permiten al servidor web identificar a los usuarios de la aplicación, así como restringir el acceso a ciertas funciones según algún criterio definido. Django es un ejemplo de framework que proporciona acceso basado en roles a la páginas, además proporciona una interfaz basada en web para crear usuarios y asignarles roles.
- **Acceso a la base de datos:** muchos frameworks cuentan con una API unificada que permite a las aplicaciones trabajar con una gran variedad de bases de datos sin necesidad de hacer cambios en el código y permitiendo a los programadores trabajar en términos de alto nivel. Además, algunos frameworks orientados a objetos contienen herramientas que proporcionan ORM (Object-Relational Mapping) que permite relacionar objetos con tuplas. Algunos frameworks además pueden proporcionar soporte transaccional y herramientas para la simplificar la migración de bases de datos.
- **Abstracción de URLs:** un framework con abstracción de URLs contiene un mecanismo para interpretarlas. En algunos casos relacionan la URL proporcionada con patrones predeterminados usando expresiones regulares, mientras que otros reescriben la URL proporcionada en otra que pueda ser reconocida por el motor. Estos mecanismos hacen que las URLs sean más fáciles de recordar y manejar por el usuario, aumentando la simplicidad del sitio y mejorando su indexación en los motores de búsqueda.
- **Sistema de plantillas web:** el objetivo de estos sistemas es reducir la cantidad de páginas en un sitio web mediante el uso de plantillas con variables que cambian de valor de manera dinámica.
- **Web caché:** algunos frameworks proporcionan mecanismos para almacenar documentos y así evitar ciertas etapas de la preparación de la página, como el acceso a la base de datos o la interpretación de la plantilla.

La mayoría de frameworks web se encargan de ofrecer una capa de controladores de acuerdo con el patrón MVC, ofreciendo mecanismos para facilitar la integración con otras herramientas para la implementación de las capas de negocio y presentación.

Puesto que el proyecto a desarrollar es una aplicación web que debido a sus características seguirá el patrón MVC, vamos a analizar dos de los principales frameworks que facilitan la implementación acorde con dicho patrón, Apache Struts y Java Server Faces (JSF). El motivo

por el que se ha elegido Apache Struts es que es un framework consolidado que facilita el desarrollo de aplicaciones web en todo tipo de plataformas compatibles con Java Enterprise, además su carácter de software libre lo convierte en una herramienta altamente disponible. La elección de JSF se debe a que surge como sustituto del primero, nutriéndose de su experiencia y tratando de mejorar algunas de sus deficiencias por lo que resulta interesante analizarlo y estudiarlo en comparación con Apache Struts.

Apache Struts

Apache Struts (desde ahora, Struts) es un framework en código libre para desarrollar aplicaciones web, siguiendo el patrón de arquitectura Modelo-Vista-Controlador [10]. Struts incluye varios mecanismos para ensamblar el componente correspondiente al controlador de las aplicaciones, es el caso de un servlet central (ActionServlet) y de varios manejadores de peticiones definidos por el desarrollador. Struts viene con una librería de etiquetas JSP UI que da soporte a la vista. La vista de una aplicación desarrollada con Struts puede hacer uso de JSTL, XSLT o Tiles entre otras tecnologías. El modelo de la aplicación se desarrolla con Java beans, mediante una gran de tecnologías como pueda ser JDBC o EJB. Todos estos conceptos serán explicados con detalle a lo largo de esta sección para comprender mejor cómo Struts implementa el patrón MVC, ver Figura 3 .

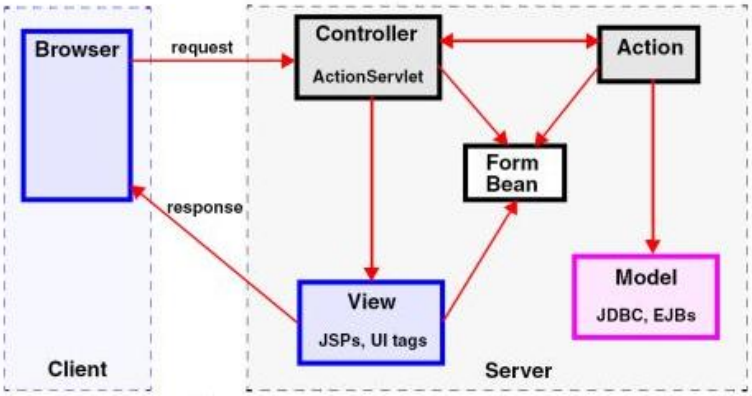

Figura 3: Desglose de la arquitectura de Struts

Struts introduce el concepto de acciones, que son usadas como gestores de peticiones cuando una URI especial es usada para la presentación de formularios. Estas acciones son las responsables de procesar los datos de formularios (beans) y redirigir a una nueva página JSP. Estas acciones funcionan como intermediarias entre el ActionServlet y el modelo de la aplicación.

Componentes del modelo

Los componentes de modelo como ya se ha explicado en el apartado de arquitectura representan la lógica de negocio y los datos que maneja la aplicación. El framework de Struts no ofrece demasiado a la hora de construir los componentes del modelo, pero probablemente es la mejor opción, ya que existen muchos frameworks disponibles para tratar con la parte del dominio de negocio de una aplicación, entre ellos Enterprise JavaBeans (EJB) y Java Data Objects (JDO). EJB es un API que proporciona un modelo de componentes distribuido estándar del lado del servidor, dotando al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación web (concurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí [8]. JDO es un API estándar basado en interfaz cuya finalidad es proporcionar al programador herramientas para

almacenar las instancias del modelo en la base de datos [11]. La ventaja de Struts es que no se limita a una única implementación de un modelo en particular.

Otra ventaja a considerar de Struts en cuanto al componente del modelo es que exige una estricta separación del código, lo que hace qué este se encuentre centralizado, haciéndolo mucho más legible y fácil de actualizar y por tanto de mantener.

Componentes de la vista

La vista en la arquitectura MVC, consiste en crear la capa de presentación que se le muestra al usuario. De manera general, los componentes de la vista en Struts se construyen a partir de páginas JSP, simplificando la creación de las mismas de dos formas:

- Separación de lógica compleja y scripts Java de las páginas JSP y moviéndolos a componentes de la vista y el controlador.
- Proporciona una serie de etiquetas que extienden la funcionalidad de JSP sin la necesidad de usar scripts Java.

Para evitar que los desarrolladores de código embeban lógica de negocio y bucles directamente en las páginas JSP, Struts proporciona una serie de etiquetas que se usan para construir los compontes de la vista, de manera que se respete la estructura del patrón MVC, estas etiquetas se agrupan en cinco librerías, de acuerdo con su funcionalidad:

- **HTML:** sus etiquetas tienen relación una a una con las etiquetas <form> de HTML. Permiten conectar el componente vista con el controlador.
- **Bean:** sus etiquetas están relacionadas con la escritura de texto, evitan que los scriptlets muestren el contenido de los objetos almacenados en la petición o en la sesión y son necesarias para usar la función de internacionalización de Struts. Se entiende por internacionalización el soporte para varios idiomas y zonas que ofrece Struts.
- **Logic:** sus etiquetas ayudan en el procesamiento condicional y la gestión de bucles, sustituyendo a los scriptlets.
- **Nested:** posee etiquetas para la visualización de propiedades anidadas dentro de un formulario u objeto.
- **Titles:** contiene etiquetas que permiten crear las distintas formas de ordenar elementos en un JSP.

Para el manejo de formularios, Struts proporciona una clase llamada ActionForm que se debe extender para darle funcionalidad. Dentro de la subclase, por cada campo dentro del formulario, se debe declarar una variable instancia con el mismo nombre que se asignó al atributo name en el código HTML, para cada variable se deben escribir métodos get y set. Adicionalmente se pueden sobrescribir los métodos reset y valídate que tienen funcionalidades específicas. El método reset es invocado para reinicializar las variables a un valor dado por el programador, mientras que el método valídate se usa para validar que los datos no tengan errores de captura o valores nulos.

Componentes del controlador

Los componentes que conforman el controlador dirigen toda la acción: gestión de formularios, navegación, etc. Además se encarga de recolectar los datos del modelo para enviárselos a la vista y que los muestren. En Struts, el componente del controlador desarrolla las siguientes actividades:

- Comprueba que los datos introducidos por el usuario sean válidos.
- Toma decisiones sobre qué componentes del modelo deben ser accedidos o actualizados.
- Recolecta los datos que la vista necesita mostrar.
- Toma decisiones sobre cómo recuperarse cuando se producen errores durante el procesamiento de una petición o respuesta.
- Decide qué componentes de la vista deben ser mostrados al usuario.

El componente central que desempeña los mecanismos de control en Struts es la clase ActionServlet. Se encarga de gestionar el flujo de control desde el fichero de configuración, los recursos internos de envío de mensajes y las fuentes de datos definidas en la aplicación. La clase ActionServlet debe ser inicializada, una vez hecho esto, estará lista para recibir peticiones de usuario. Para cada petición que reciba, los pasos que seguirá serán los siguientes:

- Identifica la acción a invocar a partir de la petición recibida.
- Llama al método perform de dicha acción si la acción solicitada ya estaba instanciada. En caso contrario, la clase Action necesaria es cargada, instanciada y almacenada en caché.
- Instancia un bean ActionForm si el método perform lo solicita. Dicho bean será inicializado con los valores introducidos por el usuario.

Estos tres pasos son ejecutados por el ActionServlet en respuesta a los comandos GET y POST de manera indistinta.

En resumen, Struts presenta numerosas ventajas a la hora de facilitar el desarrollo de aplicaciones web, destacando el transporte automático de los datos introducidos por el cliente (JSP) hasta el controlador (ActionServlet) mediante formularios (ActionForm). Además implementa la parte común a todas las aplicaciones en la parte del controlador (ActionServlet), siendo la parte particular de cada aplicación fácilmente configurable (struts-config.xml). Por último, la separación de los componentes en capas (MVC) simplifica notablemente el desarrollo y facilita su posterior mantenimiento.

Java Server Faces

Java Server Faces (JSF) es un framework para el desarrollo de aplicaciones web en Java [31]. Al igual que Struts, JSF sigue el patrón de arquitectura Modelo-Vista-Controlador. Los distintos componentes de JSF pueden agruparse siguiendo la estructura del patrón MVC, como puede verse en la Figura 4.

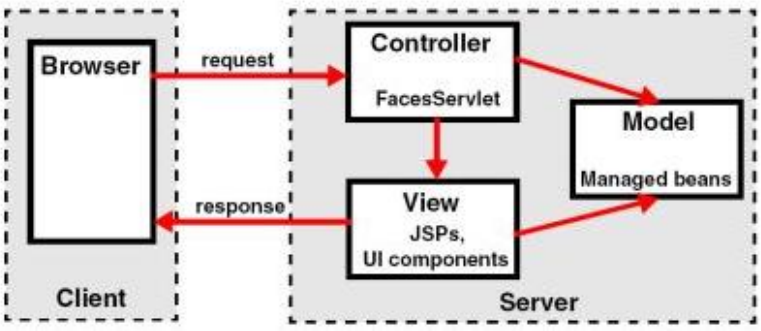

Figura 4: Desglose de la arquitectura de JSF

En JSF la gestión de beans (managed beans) representa el modelo de la aplicación. Estos Java Beans interactúan con la lógica de negocio o bien con sistemas externos como pueda ser una base de datos. La vista de la aplicación la componen distintos JSPs que son creados mediante la combinación del modelo de datos con componentes de interfaz de usuario que pueden ser predefinidos o desarrollados a medida del cliente. El FacesServlet, que dirige la navegación y gestión de objetos representa el controlador de la aplicación. Los oyentes de eventos (event listeners) también contribuyen a la lógica de control.

Componentes del modelo

Los componentes principales que forman parte del modelo de la aplicación son los managed beans que interactúan con la lógica de negocio y/o los sistemas externos; en ellos se encapsulan las interfaces de la lógica de negocio por parte de los desarrolladores. Todos y cada uno de los managed beans deben declararse en el fichero faces-config.xml, ver Figura 5, que posteriormente debe ser registrado en una aplicación JSF. Los managed beans son instanciados en tiempo de ejecución, si es necesario se inicializan y se destruyen cuando dejan de ser útiles. Según el valor que se les asigne en el elemento <managed-bean-scope> durante su declaración, los managed beans pueden ser almacenados en los siguientes ámbitos:

- **Petición (request):** el bean se almacenará en el HTTPServletRequest y será persistente entre dos páginas o la recarga de una página
- **Sesión (sesión):** el bean es almacenado en HTTPSession y existe mientras dure la sesión de usuario.
- **Aplicación (application):** el bean es almacenado en el "ServletContext" y existe durante el ciclo de vida de la aplicación JSF, con una única instancia para todos los usuarios.
- **Ninguno (none):** en este caso el bean no se almacena.

```
<managed-bean>
   <description>data record</description>
   <managed-bean-name>testData</managed-bean-name>
   <managed-bean-class>itso.jsf.TestData</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
   <managed-property>
      <property-name>property1</property-name>
      <value>test1</value>
   </managed-property>
   <managed-property>
      <property-name>property2</property-name>
      <value>test2</value>
   </managed-property>
</managed-bean>
```

Figura 5: Ejemplo de declaración de managed beans

La etiqueta <managed-bean-name> declara el nombre del bean para que pueda ser consultado en tiempo de ejecución. La etiqueta <managed-bean-class> se refiere a la clase de Java que define este managed bean.

Componentes de la Vista

Los componentes de la interfaz de usuario (UI Components) son elementos individuales representan la vista de una aplicación JSF. Estos componentes representan el estado y el comportamiento de una amplia variedad de elementos funcionales de interfaz de usuario, desde los más básicos como puedan ser un botón o un campo de texto, a otros más complejos como pueda ser una rejilla de presentación formada a partir de otros componentes. La

arquitectura de componentes de interfaz de usuario es muy flexible porque el "renderer" está separado del componente en sí, permitiendo la independencia de plataforma.

JSF además proporciona un framework oyente de eventos que permite a los gestores del lado del servidor estar conectados directamente a los eventos de interfaz de usuario. Cuando se genera un evento, éste almacena una referencia al componente que lo ha originado. La fase en la que el evento se gestionará debe ser especificada. JSF proporciona de manera estándar en sus componentes de interfaz de usuario dos eventos y sus oyentes: ActionEvent y ValueChangeEvent. Este framework simplifica de manera significativa el proceso de responder a los eventos del cliente en código del servidor, evitando el proceso de traducir complicadas peticiones HTTP.

Componentes del controlador

El componente principal de la parte del controlador en una aplicación JSF es el FacesServlet, que se puede considerar el corazón de la aplicación. Es el responsable de procesar y usar los metadatos contenidos en el fichero faces-config.xml para gestionar la navegación y el ciclo de vida del modelo. Recibe eventos de la interfaz de usuario y los relaciona con la lógica de negocio apropiada. El FacesServlet procesa las entradas externas (peticiones), durante una serie de fases, que normalmente generan una nueva salida o una respuesta. Las diferentes fases dictaminan el flujo de información dentro de la aplicación y la secuencia de eventos más comunes, ver Figura 6.

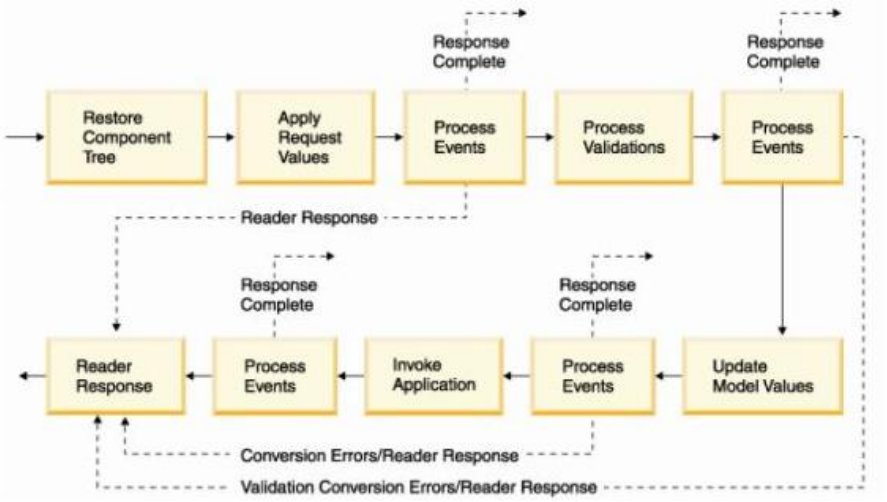

Figura 6: Ciclo de vida de JSF

A continuación se van a explicar brevemente cada una de estas fases para ayudar a comprender mejor el flujo que sigue la información, así como la importancia que tiene el FacesServlet dentro de todo el proceso.

- **Restore Component Tree (restauración del árbol de componentes):** esta fase es la responsable de reconstruir la representación de la vista en el lado del servidor in la página que se acaba de enviar. La misión del servlet es identificar la página analizando la URL.
- **Apply Request Values (aplicación de los valores solicitados):** el propósito de esta fase es provocar que se actualicen automáticamente todos los componentes de la vista.
- **Process Validations (procesamiento de validaciones):** en esta etapa cualquier validación que hayas sido registrada en algún componente por el autor de la página es invocada.

- **Update Model Values (actualización de los valores del modelo):** en esta fase todos los componentes actualizan el modelo con los datos de su nuevo estado. Al final de esta fase el modelo debería estar actualizado con todos los nuevos valores de la vista.
- **Invoke Aplication (invocar a la aplicación):** durante esta fase se manejan los eventos a nivel de aplicación invocando la lógica de negocio escrita por el desarrollador de la aplicación.
- **Render Response (carga de la respuesta):** fase final del proceso donde la respuesta se envía de vuelta al cliente y se almacena el estado de la vista por si fuera necesario usarlo en un futuro.

Comparativa entre Struts y JSF

Aunque Struts y JSF comparten bastante características y tienen una estructura similar, entre ambos frameworks existen ciertas diferencias que conviene tener en cuenta a la hora de decidirse por uno u otro según los requisitos de la aplicación. Además, también se deben tener en cuenta la capacidad de sus herramientas correspondientes, la madurez de los mismos y las direcciones futuras que se les plantean [31]. La Tabla 2 muestra la comparativa entre ambos frameworks según sus características.

	Java Server Faces	Struts
Componentes	Los componentes de interfaz de usuario se relacionan con los datos a través de eventos Componentes a medida	Librería de etiquetas específicas para Struts Se proporcionan componentes "form bean" muy básicos
Independencia de dispositivo	Kits de lectura que proporcionan independencia de dispositivo	Ninguna
Validación y gestión de errores	Framework de validación Numerosos validadores predefinidos	Validación desarrollada por un descriptor XML (validation.xml)
Navegación entre páginas	Fichero de navegación sencillo (faces-config.xml) Basado en XML	Framework sofisticado y flexible Basado en XML
Gestión de sesión y objetos	Automática	Manual
Herramientas	Cuenta con el soporte de IBM y Sun	Cuenta con un soporte IDE extenso y maduro
Madurez	Relativamente novedoso	Bastante maduro, estable y no sujeto a cambios significativos
Planes futuros	Presenta gran vitalidad. Hay implementaciones futuras planeadas, librerías de componentes aún más extensibles y nuevos contextos	Relativamente estáticos.

Tabla 2: Comparativa entre Struts y JSF

JSF fue construido siguiendo un modelo de componentes para facilitar a los desarrolladores seguir un desarrollo RAD (Rapid Application Development), mientras que Struts no sigue dicho modelo. Aunque Struts proporciona librerías de etiquetas a medida para añadirlas en Action Forms y ofrece ciertas utilidades para facilitar el trabajo de desarrollo, se orienta hacia un enfoque centrado en JSP y HTTP. JSF, en cambio, ofrece la posibilidad de construir componentes a partir de una gran variedad de tecnologías, simplificando el trabajo. En este sentido, se puede considerar JSF superior a Struts.

JSF presenta características que ofrecen independencia de dispositivo del lado del cliente. Los desarrolladores pueden usar varios kits de presentación con JSF para determinar, como debería mostrarse la aplicación en un determinado navegador o en un dispositivo portátil, etc. Struts, sin embargo, no presenta ningún mecanismo que permita desarrollar aplicaciones independientes de dispositivo. Así mismo, JSF ofrece un gran soporte para validar los datos de entrada, siendo posible usar los validadores que JSF ofrece por defecto o bien escribir validadores propios para los métodos y las clases. Cuando el valor de entrada no es aceptado se genera un mensaje de error que puede ser mostrado al usuario para que corrija sus datos de entrada. Struts proporciona dos formas para validar las propiedades de los formularios, una mediante métodos Java y otra a través de reglas definidas en un fichero XML.

La navegación es una de las características más importantes tanto en Struts como en JSF, ambos frameworks presentan un modelo de navegación declarativa y definen la navegación usando reglas a través de un fichero de configuración XML. Existen dos tipos de navegación: estática, cuando una página dirige automáticamente a otra, y dinámica, cuando alguna acción determina a qué página ir. Tanto JSF como Struts soportan ambos tipos de navegación. Aunque ambos frameworks son muy flexibles en cuanto a navegación, sin embargo, JSF simplifica el proceso ya que permite tener en una misma página varias reglas de navegación sin tener que recurrir a lógica condicional dentro del código.

La gestión de sesión y objetos es implementada de manera automática por JSF, gestionando el estado a través de las peticiones de usuario, por lo que no es necesario definirlo o implementarlo de manera manual. Struts, sin embargo, no presenta ningún mecanismo que permita gestionar el estado de la sesión y los objetos, por lo que debe definirse de manera manual.

Struts, debido a su madurez frente a JSF cuenta con numerosas herramientas de desarrollo, e incluso ha sido puesto en práctica en numerosas aplicaciones como es el caso del servidor de aplicaciones basado en Web WebSphere. Por el contrario, JSF es más novedoso por lo que ha sido menos utilizado y probado en aplicaciones web, aún así cuenta con el apoyo de IBM, Sun e incluso el creador de Struts, Craig McClanahan.

Aunque JSF ha sido diseñado como sucesor de Struts, eso no significa que JSF suponga el total remplazo de Struts. De hecho, a la hora de decidir cuál de los dos frameworks usar para desarrollar una aplicación, esta decisión no tiene por qué ser exclusiva, sino que ambos se pueden usar de manera combinada, aunque su integración puede resultar complicada y, en algunos casos, poco recomendable. Analizadas las características de ambos frameworks, su madurez y sus planes de futuro, algunos autores [31] recomiendan seguir estas recomendaciones a la hora de decidirse por uno u otro:

- Si se va a desarrollar una nueva aplicación web desde cero, es recomendable usar JSF puro, a no ser que se tengan conocimientos previos de Struts o algunos componentes reutilizables, en ese caso se podría usar JSF combinado con Struts.

- Si existe una aplicación previa desarrollada en Struts que requiere ligeros cambios para continuar con su desarrollo, lo recomendable es seguir usando Struts.
- En caso de que exista una aplicación previa desarrollada en Struts que requiera grandes cambios en la parte de la interfaz, es recomendable usar Struts combinado con JSF.

2.3 La definición del problema

La información almacenada en el WoK se actualiza cada cierto tiempo, por lo que el LEMI debe recopilarla puntualmente y añadirla a su propia base de datos. Este es un proceso complicado y laborioso ya que el WoK sólo permite descargarse ficheros que contienen un máximo de 500 registros cada uno. Además, en gran medida los registros descargados ya están almacenados en la base de datos del LEMI y normalmente el único campo a modificar es el concerniente al número de veces que un artículo de investigación ha sido citado. En la actualidad el proceso se realiza a través de funciones y expresiones en Perl, por lo que se requiere personal cualificado con conocimientos en este lenguaje. El objetivo del LEMI es simplificar todo este proceso y universalizarlo mediante el uso de una aplicación web que permita la actualización y gestión de su propia base de datos de manera intuitiva y sencilla. Mediante el uso de una aplicación web, este proceso podría hacerse mucho más rápido por lo que la actualización de los datos contenidos en la base de datos podría efectuarse con mayor frecuencia lo que implicaría que la información almacenada sería mucho más precisa y por tanto más útil en las investigaciones a realizar.

El objetivo de este proyecto es desarrollar una aplicación web que simplifique el proceso de actualización de la información bibliográfica almacenada en la base de datos de LEMI a partir de la descargada en ficheros de 500 registros de la base de datos de la Web of Knowledge. Como ya se ha anticipado este proceso presenta una serie de limitaciones que se enumeran a continuación:

- El proceso de actualización de la base de datos de LEMI sólo puede ser llevado a cabo por personal cualificado que tenga conocimientos en lenguaje Perl, mientras que si se desarrollara el proceso a través de una aplicación web unida a un gestor de bases de datos el proceso sería accesible a un mayor número de usuarios con conocimientos básicos de informática.
- El proceso actual no muestra información alguna de cómo ha sido desarrollado, ni si se ha producido algún fallo durante el mismo y las consecuencias que pudiera suponer. Esto provoca como resultado que no se inserten correctamente todos los registros disponibles y además se desconozca cuáles de los registros no han podido ser añadidos o actualizados.
- Debido a lo laborioso que resulta actualizar la información de la base de datos, esta se actualiza anualmente, con lo que se está manejando durante cierto tiempo información que se puede considerar obsoleta y que repercute de manera negativa en las líneas de investigación que hacen uso de la misma.
- Las consultas que se ejecutan sobre la base de datos no se guardan automáticamente y no existe un mecanismo que permita recuperar el resultado de esas consultas a posteriori.

Con el fin de que el proceso de actualizar la base de datos sea mucho más rápido y accesible a cualquier usuario del grupo, se desarrollará una sencilla aplicación web que tomará como datos de entrada los registros descargados del WoK y se añadirán a través de interfaces intuitivas a la base de datos del LEMI. Además, como a nivel de investigación interesa saber cuáles han sido las producciones nuevas añadidas a la base de datos y cuáles han sido

simplemente modificadas, se mostrará el resultado de la operación, especificando el número de registros modificados, insertados y aquellos que pudieran haber ocasionado algún tipo de error en el proceso. Por otro lado, la aplicación facilitará el proceso de consultar la información contenida en la base de datos y posibilitará recuperar el resultado de esas consultas en forma de tablas que podrán ser guardadas en distintos formatos para su posterior estudio y manipulación.

Gestión de proyecto software

3 Gestión de proyecto software

Se entiende por gestión de proyecto al proceso de planteamiento, ejecución y control de un proyecto, desde su comienzo hasta su conclusión, con el fin de alcanzar el mejor resultado posible en cuanto a coste, plazos y calidad [9]. Este capítulo analizará en detalle estas cuestiones e incluirá un análisis de los riesgos que pudieran surgir durante la realización de proyecto y cómo hacerles frente. Este capítulo está planteado tal y como se gestionaría el proyecto en un caso real y no como la explicación del trabajo realizado en este PFC, el objetivo es demostrar los conocimientos adquiridos a lo largo de la carrera en cuanto a gestión de proyectos se refiere.

3.1 Alcance del proyecto

La primera tarea a la hora de gestionar un proyecto es definir su alcance, es decir, delimitar el trabajo que se ha de realizar para cumplir con los objetivos y desarrollar las tareas que se han de ejecutar. En esta sección se describirán los límites del proyecto, definiendo las tareas que lo compondrán y los recursos que serán necesarios para llevarlo a buen término.

3.1.1 Definición del proyecto

El objetivo de este proyecto es desarrollar una herramienta que permita simplificar el proceso de actualización de la base de datos del LEMI, donde se almacena información bibliográfica para su posterior estudio y análisis. La herramienta tomará como entrada los ficheros descargados del WoK, que contendrán un máximo de 500 registros cada uno. Además, la aplicación facilitará la realización de consultas sobre la información guardada en la base de datos y permitirá el almacenamiento del resultado de dichas consultas para su posterior análisis con fines investigativos.

En este proyecto el almacenamiento persistente de los datos bibliográficos cobra especial importancia, por lo que durante el desarrollo del mismo deberemos centrarnos en el proceso de inserción y posterior recuperación de la información almacenada en la base de datos. Debido a que se va a manejar un gran volumen de datos es necesario prestar atención a que las consultas se ejecuten de manera eficiente y lo más rápido posible, para evitar que el usuario se pueda desesperar hasta recibir la información solicitada. Como medio de comunicación entre el usuario y la base de datos se desarrollará una aplicación web a medida, que no va a hacer uso de aplicaciones comerciales ya existentes, sino que se va a desarrollar conforme a los requisitos del cliente, partiendo desde cero y utilizando las tecnologías más adecuadas para su desarrollo.

3.1.2 Estimación de tareas y recursos

En esta sección se van detallar los costes que supondrán el proceso de análisis y desarrollo del proyecto, desglosándolos en recursos humanos, equipamiento, consumibles, viajes y gastos adicionales. Para ello se tendrá en cuenta que el tiempo que durará este proyecto será de tres meses: junio, julio y agosto, por lo que todos los gastos se detallarán en función de esa cantidad de tiempo.

En primer lugar vamos a desglosar los gastos referentes al personal implicado en la realización de este proyecto, debido a su importancia dentro del desarrollo de la aplicación. El equipo de trabajo está compuesto por cuatro miembros, cada uno de ellos con un puesto diferente y un coste por hora dependiente del mismo. Los costes por hora incluyen además el porcentaje correspondiente a lo que la empresa debe de pagar en concepto de Seguridad Social por trabajador. El coste por hora de cada rol se puede ver detallado en la Tabla 3.

	Coste/hora	Seguridad Social	Coste/hora total
Jefe de proyecto	40 €	33 %	**53,20 €**
Analista del sistema	35 €	33 %	**46,55 €**
Programador	30 €	33 %	**39,90 €**

Tabla 3: Desglose de coste por hora según rol del trabajador

Una vez indicado el coste por hora que supone cada miembro del equipo de trabajo, es necesario estimar la cantidad de días de trabajo que supone cada una de las tareas en que se divide el proyecto para cada uno de los miembros, considerando jornadas laborales de 8 horas. Este desglose de tiempo estimado por miembro del equipo en cada fase puede verse en la Tabla 4.

Fases	Jefe de proyecto	Analista del sistema	Programador 1	Programador 2	Total/ Fase
Estudio preliminar	4 días	1 día	0 días	0 días	**5 días**
Análisis del sistema	2 días	4 días	0 días	0 días	**6 días**
Diseño del sistema	0 días	4 días	0 días	0 días	**4 días**
Desarrollo de la aplicación	0 días	0 días	11 días	13 días	**24 días**
Pruebas	0 días	10 días	6 días	6 días	**22 días**
Despliegue del sistema	1 día	1 día	0 días	0 días	**2 días**
Documentación	1 día	5 días	0 días	0 días	**6 días**
Total/Rol	**8 días**	**25 días**	**17 días**	**19 días**	**69 días**

Tabla 4: Desglose de días dedicados a cada fase por trabajador

Una vez estimados los días que implica cada tarea es necesario calcular el coste que suponen en total los gastos de personal para el análisis y desarrollo del proyecto considerando jornadas laborales de 8 horas, los cuales pueden verse en la Tabla 5.

	Total días	Total horas	Coste/hora	Coste total
Jefe de proyecto	8	64	53,20 €	3.404,80 €
Analista del sistema	25	200	46,55 €	9.310,00 €
Programador 1	17	136	39,90 €	5.426,40 €
Programador 2	19	152	39,90 €	6.064,80 €
Total	69	552	-	**24.206,00 €**

Tabla 5: Desglose del coste del proyecto en personal

Tras estimar los gastos en personal es necesario analizar los gastos que va a suponer el equipamiento para el desarrollo del proyecto. Estos gastos se dividen en tres grupos, hardware, software y gastos asociados al puesto de trabajo.

Los gastos de hardware suponen el uso de cuatro puestos de trabajo para cada uno de los miembros del equipo, además se necesitará una impresora para generar la documentación asociada al proyecto y un servidor.

	Unidades	Precio unidad	Años amortización	Tiempo	Coste total
Ordenadores	4	960 €	3	3 meses	320 €
Impresora	1	250 €	3	3 meses	21 €
Servidor	1	1.200 €	-	-	1.200 €
Total	-	-	-	-	**2.421 €**

Tabla 6: Desglose del coste en hardware

En cuanto a los gastos en software, aunque la mayoría del software a emplear será de carácter open source (código libre), será necesaria la compra de tres licencias: sistema operativo Windows XP, paquete MS Office y MS Project. Los recursos open source a utilizar son el entorno de desarrollo Eclipse, el gestor de bases de datos MySQL, JavaEE como lenguaje de programación, el framework de desarrollo JSF y como servidor de aplicaciones para el despliegue JBoss.

Licencia	Coste
Windows XP	479 €
Ms Office	429 €
Ms Project	673 €
Total	**1.581 €**

Tabla 7: Desglose del coste en software

Para elaborar el presupuesto final también hay que tener en cuenta los gastos asociados a cada puesto de trabajo, incluyendo las partes proporcionales de comunicaciones, gastos de consumibles y gastos adicionales.

Descripción	Coste proporcional a un puesto	Número de puestos	Coste
Material consumible	56,25 €	4	225 €
Gastos comunicación (telefonía, Internet)	75 €	4	300 €
Gastos adicionales (alquiler, electricidad, mantenimiento, etc.)	225 €	4	900 €

Total	356,25 €	-	**1.425 €**

Tabla 8: Desglose coste asociado a los puestos de trabajo

3.1.3 Presupuesto

En el presupuesto final del proyecto se incluyen además otros gastos, los cuales se detallan a continuación:

- **Riesgos:** se ha añadido al presupuesto final un 10% extra para hacer frente a los posibles riesgos que pudieran surgir durante el desarrollo del proyecto.
- **Beneficios:** se ha incrementado el presupuesto en un 10%, que representan los beneficios tras la finalización del proyecto.
- **IVA:** se ha considerado además, el 16% de IVA reglamentario.

Así, el presupuesto total sin IVA del presente proyecto asciende a 35.559,60 € (treinta y cinco mil quinientos cincuenta y nueve euros y sesenta céntimos). El presupuesto final, incluyendo el IVA y que será el total que deberá pagar el cliente a la empresa asciende a **41.249,14 €** (cuarenta y un mil doscientos cuarenta y nueve euros y catorce céntimos). El desglose completo de presupuesto final puede apreciarse en la Tabla 9.

Descripción	Coste
Gastos personal	24.206,00 €
Gastos hardware	2.421,00 €
Gastos software	1.581,00 €
Gastos puesto de trabajo	1.425,00 €
Total costes directos	**29.633,00 €**
Riesgos (10%)	2.963,30 €
Beneficios (10%)	2.963,30 €
Total sin IVA	**35.559,60 €**
IVA (16%)	5.689,54 €
Total a pagar	**41.249,14 €**

Tabla 9: Presupuesto total del proyecto

3.2 Plan de trabajo

Una vez acotado el alcance del proyecto y el presupuesto que supone su realización es necesario identificar las tareas en las que éste se va a dividir para hacer una estimación de su duración y poder hacer una planificación adecuada de las mismas, con el fin de respetar los costes y plazo acordados con el cliente.

3.2.1 Identificación de tareas

Una vez analizada la naturaleza del proyecto y delimitado su alcance, se ha considerado apropiado dividirlo en las siguientes tareas:

- **Estudio preliminar del sistema:** esta tarea comienza con la detección del problema que nuestra aplicación pretende resolver, para continuar con el esbozo de la solución al mismo y estimar la planificación que se deberá seguir durante su desarrollo. Por lo tanto, esta primera fase puede dividirse en dos partes:
  - **Planteamiento del problema y esbozo de la solución:** en esta parte se detecta cuál es el problema que plantea la necesidad de desarrollar una aplicación que lo resuelva y se define de qué manera la aplicación lo va a solucionar.
  - **Elaboración del plan de proyecto:** tras detectar el problema y plantear una posible solución es necesario decidir cómo se va a gestionar el tiempo y los recursos disponibles para su realización, es por tanto en esta fase cuando se detalla la planificación a seguir, las tareas en que se dividirá el proyecto y cuál será el coste que supondrá el desarrollo completo del proyecto.
- **Análisis del sistema:** el objetivo de esta fase es identificar las necesidades del cliente, así como su viabilidad, esto se consigue mediante la extracción de requisitos, que se realiza en dos fases:
  - **Definición de requisitos:** consiste en determinar los requisitos generales que debe satisfacer la aplicación mediante sesiones de trabajo con el cliente.
  - **Especificación de requisitos:** en esta tarea se describe de manera completa el comportamiento del sistema a desarrollar, mediante el estudio en detalle de cada uno de los requisitos previamente definidos.
- **Diseño del sistema:** esta fase se encarga de desarrollar las directrices propuestas durante el análisis en función de la configuración que tenga más posibilidades de satisfacer los objetivos planteados.
  - **Diseño arquitectónico:** en esta tarea se definen las relaciones entre todos y cada uno de los elementos estructurales del programa.
  - **Diseño del modelo de datos:** consiste en transformar el modelo de dominio de la información que se ha creado durante la fase de análisis en la estructuras de datos necesarios para implementar la aplicación.
  - **Diseño de la lógica de negocio:** se define la parte del sistema que no ve el usuario y que especifica las rutinas de procesamiento de datos.
  - **Diseño de la interfaz:** describe como se comunica la aplicación consigo misma, qué sistemas operan en colaboración con ella y los operadores y usuarios que la emplearán.
- **Desarrollo de la aplicación:** en esta fase se generará el código necesario para desarrollar la aplicación y que funcione cumpliendo con los objetivos marcados en la fase de análisis. Se puede dividir en las siguiente partes:
  - **Desarrollo de la interfaz:** se producirán los archivos necesarios para generar la capa de presentación que será vista por el usuario y con la que interactuará.
  - **Desarrollo del dominio:** se generará el código necesario para representar la información del sistema.
  - **Desarrollo de la lógica de negocio:** se desarrollará el código necesario para gestionar las peticiones recibidas desde la interfaz y que requieran el procesamiento de datos.

  - **Desarrollo del proceso de inserción de datos:** se generará el código asociado al proceso de inserción de datos por parte del usuario, incluyendo todas las comprobaciones necesarias para posteriormente generar la información sobre el desarrollo del proceso.
  - **Generación de tablas con los resultados:** desarrollo de las clases necesarias para mostrar de manera ordena en tablas los datos fruto de una consulta o de una petición por parte del usuario.
- **Pruebas:** en esta fase se definirán y ejecutarán diferentes pruebas para comprobar el correcto funcionamiento de la aplicación de acuerdo con los objetivos que debe cumplir. Se puede dividir en tres tareas:
  - Definición de pruebas unitarias: se definen distintas pruebas para comprobar que los distintos componentes del sistema funcionan correctamente por separado.
  - Definición de pruebas de integración: su función es asegurar el correcto funcionamiento del sistema en su totalidad.
  - Ejecución de pruebas: tras la definición de las distintas pruebas es necesario ejecutarlas para comprobar que el sistema realmente funciona.
- **Despliegue del sistema:** esta fase está formada por una única tarea en si misma que corresponde a la generación de los elementos necesarios para la implantación del sistema.
- **Documentación:** esta tarea consiste en la generación de todos los documentos asociados a la aplicación, incluyendo desde el análisis y diseño de la misma hasta el documento que recoge el resultado de las pruebas ejecutadas y el manual de usuario.

3.2.2 Estimación de tareas

Una vez definidas las tareas en que se va a dividir el proyecto, es necesario estimar los días que va a llevar cada una de las tareas y fases, para hacernos una idea más precisa de la envergadura del proyecto. La duración estimada que cada una de las tareas puede verse detallada en la Tabla 10.

Fase	Tarea	Días estimados
Fase de estudio preliminar	Planteamiento del problema y realización de oferta al cliente.	3 días
	Elaboración del plan de proyecto.	2 días
	Total fase de estudio preliminar	5 días
Fase de análisis del sistema	Definición de requisitos	4 días
	Especificación de requisitos	2 días
	Total fase de análisis	6 días
Fase de diseño del sistema	Diseño de la arquitectura	1 día
	Diseño del modelo de datos	1 día
	Diseño de la lógica de negocio	1 día
	Diseño de la interfaz	1 día
	Total fase de diseño	4 días
Fase de desarrollo de la aplicación	Desarrollo de la interfaz	5 días
	Desarrollo del dominio	2 días
	Desarrollo de la lógica de negocio	5 días
	Desarrollo del proceso de inserción de datos	7 días
	Generación de tablas con los resultados	5días
	Total fase de desarrollo	24 días

Fase de pruebas	Definición de pruebas unitarias	5 días
	Definición de pruebas de integración	5 días
	Ejecución de pruebas	12 días
	Total fase de pruebas	22 días
Fase de despliegue	Despliegue del sistema	2 días
	Total fase de despliegue	2 días
Fase de documentación	Documentación fase de planificación	1 día
	Documentación fase de análisis	1 día
	Documentación fase de diseño	1 día
	Documentación fase de construcción	1 día
	Documentación fase de implantación	1 día
	Manual de usuario	1 día
	Total fase de documentación	6 días

Tabla 10: Duración estimada por tareas y fases

3.2.3 Planificación de tareas

Una vez identificadas y estimadas las tareas en que se dividirá el proyecto es necesario hacer una planificación para ver cómo se van a distribuir dichas tareas en el eje del tiempo y poder ver las dependencias existentes entre ellas. Es importante tener en cuenta que esta planificación estimada para un grupo de trabajo de cuatro personas no corresponde con la planificación real del proyecto fin de carrera, desarrollada por una única persona y que puede en los anexos que aparecen al final de este documento.

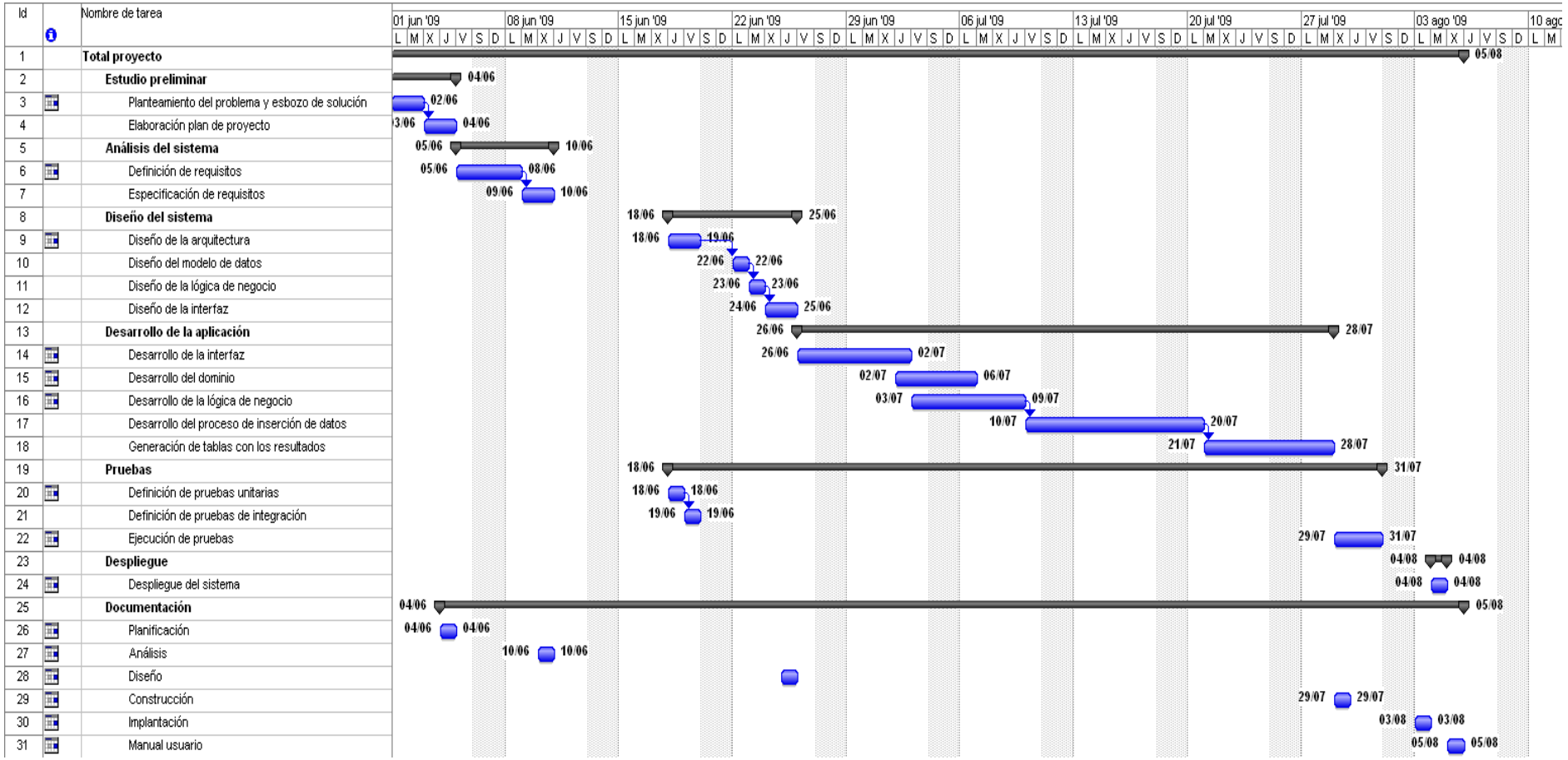

Figura 7: Planificación del proyecto

3.3 Gestión de recursos

Debido a que en el apartado 3.1.2 de estimación de tareas y recursos ya se ha hablado de los recursos que se van a necesitar para el desarrollo de este proyecto, tanto en cuestión de personal, como de equipamiento, consumibles, viajes y gastos adicionales, en esta sección nos vamos a centrar simplemente en los recursos humanos, ya que suponen la base para el desarrollo de la aplicación.

Para este proyecto se ha considerado un equipo de trabajo compuesto por cuatro miembros, compuesto por un jefe de proyecto, un analista y dos programadores. En la siguiente sección se va a entrar en detalle sobre qué características deben ser satisfechas para cubrir cada uno de los roles.

3.3.1 Especificación de recursos

Como ya se ha comentado, el equipo de trabajo está compuesto por cuatro miembros. No se ha considerado necesario un número mayor de miembros, ya que no se trata de un proyecto de gran envergadura y de esta forma se garantiza una mejor comunicación entre los miembros del equipo. Tampoco es aconsejable reducir el número de miembros que forman parte de equipo, ya que de esta forma cada rol está especializado en una serie de tareas distintas y por tanto las pueden realizar de una manera más eficaz y profesional. La Figura 8 muestra un organigrama con la organización del equipo de trabajo, se puede ver que se trata de una organización de tipo jerárquico donde la parte más alta la ocupa el cliente, ya que es la persona más importante en el desarrollo del sistema y son sus necesidades las que el sistema debe cubrir. Por debajo del cliente se encuentra el jefe de proyecto, debido a que es el encargado de coordinar, dirigir y controlar al resto de los integrantes del equipo de trabajo. Los puestos de analista y programador se encuentran al mismo nivel ya que tratan áreas diferentes.

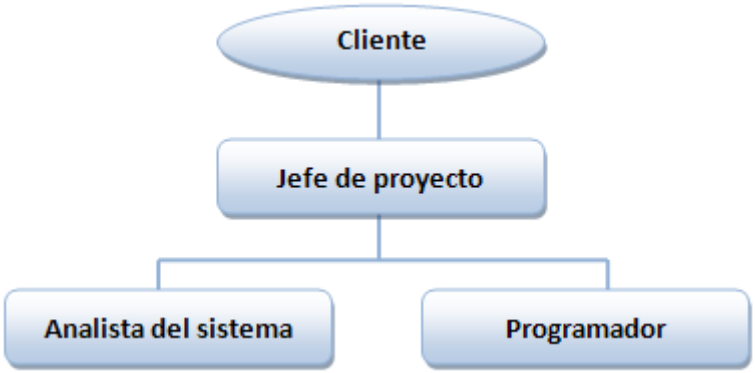

Figura 8: Organigrama del equipo de trabajo

A continuación se van a describir las características que deben de cumplir cada uno de los miembros del equipo de trabajo de acuerdo con el puesto que desempeñen dentro del mismo:

- **Jefe de proyecto:** debe contar con una excelente capacidad para coordinar y dirigir a los diferentes miembros del equipo, haciendo uso de sus dotes de mando y fomentando una cordial relación con el resto del personal. Debido a que será el responsable del proyecto, deberá encargarse de mantener una estrecha relación con el cliente para conseguir los mejores resultados.
- **Analista del sistema:** se busca una persona con una alta capacidad de abstracción y análisis. Debe ser además un buen interlocutor y con excelentes habilidades sociales, ya que debe tratar a menudo con el cliente con el fin de recoger los

requisitos que fundamentan el sistema. También debería contar con una dilatada experiencia en el campo del diseño con el fin de diseñar un sistema eficiente y fácil de usar.

- **Programador:** se precisa incorporar dos programadores al equipo de trabajo que deben dominar las tecnologías y herramientas elegidas para el desarrollo del sistema, tener unos amplios conocimientos de programación orientada a objetos y de gestión de bases de datos. Deben tener además capacidad para soportar la presión, facilidad para aprender nuevos lenguajes de programación y una fuerte adaptación a cambios tecnológicos.

Asignación de recursos

Una vez detallados los miembros que formarán parte del equipo de trabajo es necesario asignarles las tareas que han de realizar y estimar el tiempo que les va a suponer cada una de ellas, esta asignación puede verse en la Tabla 11.

Tarea	Jefe de proyecto	Analista del sistema	Programador 1	Programador 2	Total/ Tarea
Estudio Preliminar					
Planteamiento del problema y realización de la oferta al cliente	2 días	1 días	0 días	0 días	**3 días**
Elaboración del plan de proyecto	2 días	0 días	0 días	0 días	**2 días**
Análisis del sistema					
Definición de requisitos	2 días	2 días	0 días	0 días	**4 días**
Especificación de requisitos	0 días	2 días	0 días	0 días	**2 días**
Diseño del sistema					
Diseño de la arquitectura	0 días	1 día	0 días	0 días	**1 día**
Diseño del modelo de datos	0 días	1 día	0 días	0 días	**1 día**
Diseño de la lógica de negocio	0 días	1 día	0 días	0 días	**1 día**
Diseño de la interfaz	0 días	1 día	0 días	0 días	**1 día**
Desarrollo de la aplicación					
Desarrollo de la interfaz	0 días	0 días	2 días	3 días	**5 días**

Desarrollo del dominio	0 días	0 días	1 día	1 día	**2 días**
Desarrollo de la lógica de negocio	0 días	0 días	3 días	2 días	**5 días**
Desarrollo del proceso de inserción de datos	0 días	0 días	3 días	4 días	**7 días**
Generación de tablas con los resultados	0 días	0 días	2 días	3 días	**5 días**
Pruebas					
Definición pruebas unitarias	0 días	3 días	1 día	1 día	**5 días**
Definición pruebas de integración	0 días	3 días	1 día	1 día	**5 días**
Ejecución de pruebas	0 días	4 días	4 días	4 días	**12 días**
Despliegue					
Despliegue del sistema	1 día	1 día	0 días	0 días	**2 días**
Documentación					
Planificación	1 día	0 días	0 días	0 días	**1 día**
Análisis	0 días	1 día	0 días	0 días	**1 día**
Diseño	0 días	1 día	0 días	0 días	**1 día**
Construcción	0 días	1 día	0 días	0 días	**1 día**
Implantación	0 días	1 día	0 días	0 días	**1 día**
Manual usuario	0 días	1 día	0 días	0 días	**1 día**
Total (días)	**8 días**	**25 días**	**17 días**	**19 días**	**69 días**

Tabla 11: Asignación de los recursos a las tareas

3.4 Gestión de riesgos

Se entiende por riesgo en un proyecto todo tipo de eventos o condiciones inciertas que, en caso de ocurrir, tienen algún tipo de efecto sobre los objetivos del proyecto [22]. La función de la gestión de riesgos del software es identificar, estudiar y eliminar las fuentes de riesgo antes de que empiecen a amenazar la finalización satisfactoria de un proyecto software [29]. Esta es la precisamente la tarea que vamos a llevar a cabo en esta sección, identificar todas las

posibles fuentes de riesgos y analizarlas para conocer las consecuencias y cómo debería actuarse en el caso de que a lo largo del desarrollo del proyecto pudieran surgir.

3.4.1 Identificación de riesgos

En esta sección se van a tratar de especificar todas las posibles amenazas que afectan al plan de proyecto, es decir, a las estimaciones, planificación temporal, recursos, etc. Mediante la identificación de los riesgos conocidos y predecibles se da un paso adelante para evitarlos cuando sea posible y controlarlos cuando sea necesario.

Riesgo-01					
Descripción	El producto o el esfuerzo son mayores que los estimados (en líneas de código, en el número de puntos función, o en relación con el tamaño del proyecto anterior).				
Causas	Se ha producido una mala estimación durante la fase de elaboración del plan de proyecto.				
Consecuencias	Es necesario hacer una re-planificación del proyecto, para adaptarse a las necesidades reales del mismo. Puede suponer no cumplir con los plazos previstos o la necesidad de aumentar los recursos asociados al proyecto para cumplirlos.				
Probabilidad	Muy alta	Alta	**Media**	Baja	Muy baja

Tabla 12: Descripción del Riesgo-01

Riesgo-02					
Descripción	La fecha final ha cambiado sin ajustarse al ámbito del producto o a los recursos disponibles.				
Causas	La fecha final puede ser modificada por el cliente, los usuarios finales o factores externos.				
Consecuencias	Es necesario re-planificar los plazos y entregas para ajustarse a la nueva fecha final. En caso de que la fecha de entrega se anticipe puede afectar a la calidad del producto final y puede ser necesario aumentar los recursos asignados al proyecto.				
Probabilidad	Muy alta	Alta	**Media**	Baja	Muy baja

Tabla 13: Descripción del Riesgo-02

Riesgo-03					
Descripción	Las herramientas de desarrollo no se han elegido en función de sus características técnicas, y no proporcionan las prestaciones previstas.				
Causas	La fase de estudio de herramientas disponibles en el estado de la cuestión no se ha adaptado a las necesidades de la aplicación, ni se ha realizado de manera correcta.				
Consecuencias	Es necesario volver a repetir la fase de análisis y estudio de las herramientas existentes para buscar aquellas que proporcionen las prestaciones necesarias para el desarrollo de la aplicación. Esto puede generar retrasos debido a que es necesario repetir ciertas tareas y aprender a manejar nuevas herramientas.				
Probabilidad	Muy alta	Alta	Media	**Baja**	Muy baja

Tabla 14: Descripción del Riesgo-03

Riesgo-04					
Descripción	La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado.				
Causas	El personal contratado no cuenta con conocimientos suficientes en las herramientas de desarrollo a utilizar.				
Consecuencias	Es necesario emplear más tiempo del planificado en el proceso de aprendizaje, llevando a la re-planificación de otras tareas.				
Probabilidad	Muy alta	Alta	**Media**	Baja	Muy baja

Tabla 15: Descripción del Riesgo-04

Riesgo-05					
Descripción	El cliente o los usuarios finales insisten en nuevos requisitos.				
Causas	El cliente o los usuarios no tienen clara la funcionalidad que buscan en la aplicación, lo que se traduce en la incorporación continua de nuevos requisitos.				
Consecuencias	La inclusión de nuevos requisitos supone una demora en los plazos de entrega y por tanto un aumento en el presupuesto final.				
Probabilidad	Muy alta	Alta	**Media**	Baja	Muy baja

Tabla 16: Descripción del Riesgo-05

Riesgo-06					
Descripción	En el último momento, a los usuarios finales no les gusta el producto, por lo que hay que volver a diseñarlo y a construirlo.				
Causas	Una especificación incorrecta de los requisitos por parte del analista.				
Consecuencias	Es necesario rediseñar y volver a construir el producto, supone un incremento notable en el presupuesto final.				
Probabilidad	Muy alta	Alta	Media	Baja	**Muy baja**

Tabla 17: Descripción del Riesgo-06

Riesgo-07					
Descripción	El cliente no participa en los ciclos de revisión de los planes, prototipos y especificaciones, o es incapaz de hacerlo, resultando unos requisitos inestables y la necesidad de realizar unos cambios que consumen tiempo.				
Causas	El cliente no tiene interés en vincularse con el proyecto o carece del tiempo necesario para hacerlo.				
Consecuencias	Se generan unos requisitos inestables, que suponen una continua re-planificación y rediseño del sistema, consumiendo estas operaciones mucho tiempo y dificultando la entrega del sistema en el plazo previsto.				
Probabilidad	Muy alta	Alta	Media	**Baja**	Muy baja

Tabla 18: Descripción del Riesgo-07

Riesgo-08					
Descripción	Algún miembro del equipo de trabajo no suministra los componentes en el período establecido.				
Causas	Problemas de coordinación dentro del equipo de trabajo, bajas de personal, dificultades en el manejo de las tecnologías elegidas, etc.				
Consecuencias	Retrasos en las entregas acordadas con el cliente, disminución de la calidad final del producto.				
Probabilidad	Muy alta	Alta	Media	Baja	**Muy baja**

Tabla 19: Descripción del Riesgo-08

Riesgo-09					
Descripción	Algún miembro del equipo proporciona componentes o documentación de una calidad inaceptable, por lo que hay que añadir un tiempo extra para mejorar la calidad.				
Causas	El personal carece de la motivación o los conocimientos necesarios para hacer un trabajo de calidad aceptable.				
Consecuencias	Retrasos en las entregas para aumentar la calidad de los productos a entregar al cliente.				
Probabilidad	Muy alta	Alta	Media	**Baja**	Muy baja

Tabla 20: Descripción del Riesgo-09

Riesgo-010					
Descripción	Los requisitos no se han definido correctamente y su redefinición aumenta el ámbito del proyecto.				
Causas	Mala comunicación con el cliente durante la especificación de requisitos.				
Consecuencias	Aparecen problemas para cumplir con los plazos establecidos y esto se refleja en el presupuesto del proyecto.				
Probabilidad	Muy alta	Alta	**Media**	Baja	Muy baja

Tabla 21: Descripción del Riesgo-010

Riesgo-011					
Descripción	Los miembros del equipo no se implican en el proyecto, y por lo tanto no alcanzan el nivel de rendimiento deseado.				
Causas	El personal carece de la motivación necesaria para llevar a cabo el proyecto de manera eficiente.				
Consecuencias	El jefe de proyecto de buscar la forma de motivar al personal, sino los plazos de entrega se retrasan, disminuye la calidad del producto final y no es posible ceñirse al presupuesto establecido.				
Probabilidad	Muy alta	Alta	Media	**Baja**	Muy baja

Tabla 22: Descripción del Riesgo-011

Riesgo-012					
Descripción	Los conflictos entre los miembros del equipo conducen a problemas en la comunicación y en el diseño, errores en la interfaz y tener que repetir algunos trabajos.				
Causas	El jefe de proyecto no es capaz de mantener una armonía y un buen ambiente de trabajo dentro del equipo de desarrollo.				
Consecuencias	Productos de baja calidad, necesidad de repetir parte del trabajo, retrasos en la entrega de productos, aplicación que no se adapta a las necesidades del usuario.				
Probabilidad	Muy alta	Alta	Media	**Baja**	Muy baja

Tabla 23: Descripción del Riesgo-012

Riesgo-013	
Descripción	Alguien de la plantilla abandona el proyecto antes de su finalización.
Causas	Algún miembro puede abandonar el proyecto porque encuentre una oportunidad mejor en otro trabajo, no se encuentre a gusto en el proyecto o por causa de alguna enfermedad.
Consecuencias	Supone una mayor sobrecarga para el resto de miembros del equipo

	que deben repartirse las tareas asignadas al miembro ausente. Puede generar retrasos en las entregas acordadas con el cliente.				
Probabilidad	Muy alta	Alta	**Media**	Baja	Muy baja

Tabla 24: Descripción del Riesgo-013

Riesgo-014					
Descripción	Se producen pérdidas de información, bien en la documentación o en los productos desarrollados.				
Causas	No se ha guardado bien el producto perdido o bien se ha producido algún error en el soporte en el que se almacenaba.				
Consecuencias	Tener que rehacer la parte de la documentación o implementación perdida, suponiendo un retraso en las entregas y en el presupuesto. Si la pérdida es considerable, es necesario re-planificar completamente el proceso de desarrollo.				
Probabilidad	Muy alta	Alta	**Media**	Baja	Muy baja

Tabla 25: Descripción del Riesgo-014

3.4.2 Análisis de riesgos

Hasta ahora se han definido los diferentes riesgos que podrían surgir durante el análisis y desarrollo del proyecto, sin embargo, y aunque se ha hablado de posibles consecuencias de cada uno de ellos, es necesario calcular cuál es su impacto sobre la totalidad del proyecto. Para ello se van a tener en cuenta tres factores que pueden verse afectados por culpa de los riesgos: la calidad del producto final, el coste y el tiempo invertido. También es importante tener en cuenta la probabilidad existente de que ese riesgo surja, por lo tanto también se recogerá. Cada uno estos factores tomara uno de los siguientes factores: muy alto, alto, medio, bajo o muy bajo. La Tabla 26 muestra los criterios de impacto que se han tenido en cuenta en función del factor de riesgo.

	Muy bajo	**Bajo**	**Medio**	**Alto**	**Muy alto**
Coste	Supone un aumento en el coste del proyecto inferior al 5%.	Supone un aumento en el coste del proyecto entre el 5% y el 10%.	Supone un aumento en el coste del proyecto entre el 10% y el 15%.	Supone un aumento en el coste del proyecto entre el 15% y el 20%.	Supone un aumento en el coste del proyecto superior al 20%.
Tiempo	Supone un retraso total en la entrega entre un día y una semana.	Supone un retraso total en la entrega mayor de una semana e inferior a dos.	Supone un retraso total en la entrega entre dos y tres semanas.	Supone un retraso total en la entrega mayor de tres semanas e inferior a cuatro semanas.	Supone un retraso total en la entrega a partir de un mes.
Calidad	Supone un descenso de la calidad del proyecto inferior al 5%.	Supone un descenso de la calidad del proyecto entre el 5% y el 10%.	Supone un descenso de la calidad del proyecto entre el 10% y el 15%.	Supone un descenso de la calidad del proyecto entre el 15% y el 20%.	Supone un descenso de la calidad del proyecto superior al 20%.

Tabla 26: Impacto según el factor de riesgo

Una vez explicado cómo se va a medir el impacto de cada uno de los factores de riesgo es necesario analizar cada uno los riesgos identificados. El resultado de dicho análisis puede verse en la Tabla 27.

	Coste	Tiempo	Calidad	Probabilidad
Riesgo-001	Alto	Alto	Medio	Media
Riesgo-002	Medio	Medio	Alto	Media
Riesgo-003	Medio	Medio	Bajo	Baja
Riesgo-004	Medio	Alto	Medio	Media
Riesgo-005	Alto	Alto	Medio	Media
Riesgo-006	Muy alto	Muy alto	Alto	Muy baja
Riesgo-007	Alto	Alto	Alto	Baja
Riesgo-008	Alto	Alto	Alto	Muy baja
Riesgo-009	Medio	Medio	Alto	Baja
Riesgo-010	Medio	Medio	Bajo	Media
Riesgo-011	Bajo	Bajo	Media	Baja
Riesgo-012	Alta	Alta	Alta	Baja
Riesgo-013	Alta	Alta	Alta	Media
Riesgo-014	Alto	Muy alto	Medio	Media

Tabla 27: Análisis de riesgos según su impacto

Tras analizar los distintos riesgos que pueden surgir durante el desarrollo del proyecto y el impacto que pueden causar en el mismo, es necesario elaborar un plan de contingencia para prevenir su aparición, o en caso de que sea inevitable que se produzca, se deben detallar las medidas a tomar para minimizar los daños. Por cada riesgo, se van a recoger una serie de acciones a seguir que serán recogidas en una tabla con los siguientes campos:

- **Identificador:** identificador del riesgo.
- **Descripción:** descripción del riesgo para el que se van a detallar las medidas.
- **Objetivo del plan:** procesos que se van a cubrir en el riesgo especificado.
- **Prevención:** medidas que se deben tomar para evitar la aparición del riesgo.
- **Contingencia:** acciones que se deben seguir en caso de que el riesgo se produzca.

Las siguientes tablas recogen el plan de contingencia a aplicar para minimizar el impacto de los riegos asociados a este proyecto:

Riesgo-01	
Descripción	El producto o el esfuerzo son mayores que los estimados (en líneas de código, en el número de puntos función, o en relación con el tamaño del proyecto anterior).
Objetivo	Hacer una estimación los más precisa posible, que se ajuste al tamaño real del proyecto para evitar re-planificaciones y no cumplir los plazos.
Prevención	Estimar de la manera más precisa posible el tamaño del proyecto, basándonos no sólo en técnicas como los puntos función, sino también a partir de la experiencia de proyectos anteriores. También es recomendable hacer una planificación que cuente con tiempo extra que poder utilizar en caso de que la extensión del proyecto supere la estimada.
Contingencia	Hacer uso del tiempo extra que se ha incluido en la planificación para

	mitigar este riesgo. Si aún así este tiempo es insuficiente o no se ha contado con él, solicitar al equipo de trabajo que haga horas extras para cumplir los plazos o contratar a personal de apoyo.

Tabla 28: Medidas de prevención y contingencia Riesgo-01

Riesgo-02	
Descripción	La fecha final ha cambiado sin ajustarse al ámbito del producto o a los recursos disponibles.
Objetivo	Evitar que los reajustes de fecha tengan impacto en el coste y calidad del proyecto.
Prevención	Acordar una fecha razonable con el cliente que se ajuste a las necesidades del proyecto y a la planificación estimada en el plan de proyecto.
Contingencia	Explicarle al cliente las consecuencias que implica un cambio de fecha en el proyecto y pedirle que corra con los gastos que pueda ocasionar ese cambio de fecha, ya que el presupuesto está ajustado a una fecha en concreto y cualquier cambio en la misma puede hacer que el coste final varíe.

Tabla 29: Medidas de prevención y contingencia Riesgo-02

Riesgo-03	
Descripción	Las herramientas de desarrollo no se han elegido en función de sus características técnicas, y no proporcionan las prestaciones previstas.
Objetivo	Elegir herramientas y tecnologías que se ajusten a las necesidades del proyecto y que proporcionen la funcionalidad necesaria para su desarrollo.
Prevención	Dedicar tiempo suficiente en la fase de análisis a estudiar las distintas herramientas y tecnologías existentes para elegir aquellas que se adecúen a las necesidades del proyecto. Tomar como referencia aquellas usadas en proyectos de índole similar.
Contingencia	En el momento en que se detecte que las herramientas no son las adecuadas tratar de buscar otras que proporcionen las prestaciones necesarias lo antes posible para evitar seguir acumulando retrasos. Si existen varios tipos de herramientas que cumplen con la funcionalidad buscada, seleccionar aquellas cuyo aprendizaje sea más sencillo o en las que el equipo de desarrollo tenga experiencia, para minimizar el tiempo que supone el aprendizaje de las nuevas herramientas.

Tabla 30: Medidas de prevención y contingencia Riesgo-03

Riesgo-04	
Descripción	La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado.
Objetivo	Minimizar el tiempo de aprendizaje que supone la utilización de una nueva herramienta de desarrollo.
Prevención	Elegir herramientas sencillas de utilizar o aquellas donde el equipo de desarrollo tenga ya experiencia en su uso.
Contingencia	Reajustar los plazos estimados en el plan de proyecto. Pedir colaboración por parte del equipo de trabajo, para que haga un sobreesfuerzo y así evitar retrasos en las entregas del proyecto al cliente.

Tabla 31: Medidas de prevención y contingencia Riesgo-04

Riesgo-05	
Descripción	El cliente o los usuarios finales insisten en nuevos requisitos.
Objetivo	Tratar de diseñar un sistema que cumpla con todas las necesidades especificadas por el cliente, sin que esto suponga una demora en los plazos o un aumento del coste del proyecto.
Prevención	Analizar en detalle cuáles son las necesidades del cliente, incluso aquellas que no ha manifestado pero son propensas a surgir durante el desarrollo del proyecto. Fijar de antemano con el cliente la funcionalidad que ha de cumplir el sistema para evitar la inclusión de nuevos requisitos durante el desarrollo del proyecto.
Contingencia	Renegociar con el cliente los plazos de entrega y/o el presupuesto que supone la inclusión de nuevos requisitos en etapas tardías del proyecto.

Tabla 32: Medidas de prevención y contingencia Riesgo-05

Riesgo-06	
Descripción	En el último momento, a los usuarios finales no les gusta el producto, por lo que hay que volver a diseñarlo y a construirlo.
Objetivo	Evitar la completa reconstrucción del sistema una vez finalizado y entregado al cliente.
Prevención	Asegurarse de que los requisitos recogen todas y cada una de las necesidades manifestadas por el cliente. Hacer entregas parciales para conocer la opinión del cliente sobre el producto. Elaborar un prototipo que pueda dar al cliente una idea de cómo se va a desarrollar el producto para que pueda indicar qué mejoraría del mismo.
Contingencia	Explicar al cliente las sanciones que conllevaría rediseñar el producto entero, teniendo en cuenta que satisface los requisitos que él había firmado.

Tabla 33: Medidas de prevención y contingencia Riesgo-06

Riesgo-07	
Descripción	El cliente no participa en los ciclos de revisión de los planes, prototipos y especificaciones, o es incapaz de hacerlo, resultando unos requisitos inestables y la necesidad de realizar unos cambios que consumen tiempo.
Objetivo	Aumentar el interés y la participación del cliente en el proceso de desarrollo del proyecto.
Prevención	Fijar un calendario de reuniones con suficiente antelación y que el cliente lo apruebe. Asignar a un responsable de asistir a esas reuniones en caso de que el cliente no pueda. Organizar el contenido de las reuniones con antelación para que el tiempo con el cliente sea productivo. Establecer sanciones en caso de que el cliente o su responsable falten a las reuniones.
Contingencia	Aplicar las sanciones previstas en caso de que no se acuda de manera regular a las reuniones.

Tabla 34: Medidas de prevención y contingencia Riesgo-07

Riesgo-08	
Descripción	Algún miembro del equipo de trabajo no suministra los componentes en el período establecido.
Objetivo	Conseguir que todos los miembros del equipo de trabajo cumplan con los plazos de entrega asignados.

Prevención	Planificar periodos de desarrollo realistas y acordes con la capacidad de cada trabajador.
Contingencia	Reasignar el trabajo de ese miembro para que cuente con la colaboración de otros miembros con menos carga de trabajo. Si se repite de manera continuada, aplicar algún tipo de sanción.

Tabla 35: Medidas de prevención y contingencia Riesgo-08

Riesgo-09	
Descripción	Algún miembro del equipo proporciona componentes o documentación de una calidad inaceptable, por lo que hay que añadir un tiempo extra para mejorar la calidad.
Objetivo	Conseguir que todos los entregables tengan una calidad aceptable que cumpla con las exigencias del proyecto.
Prevención	Asegurarse de que todos los miembros del equipo cuentan con motivación suficiente para el desarrollo del proceso. Asignar la carga de trabajo de cada miembro de forma equitativa y realista sin sobrecargar a ninguno de ellos.
Contingencia	Mantener una conversación con el miembro del equipo en cuestión para comprobar que está motivado y conocer las razones por las que entrega el material con una calidad inaceptable. Reasignar parte de su trabajo entre el resto de componentes del equipo.

Tabla 36: Medidas de prevención y contingencia Riesgo-09

Riesgo-010	
Descripción	Los requisitos no se han definido correctamente y su redefinición aumenta el ámbito del proyecto.
Objetivo	Establecer un conjunto estable y correcto de requisitos que evite continuas redefiniciones.
Prevención	Establecer reuniones con el cliente para asegurarnos con los requisitos son los adecuados. Asegurarnos de incluir todos los requisitos necesarios de tal forma que el ámbito del proyecto quede bien delimitado.
Contingencia	Re-planificar el proyecto para incluir el tiempo que conlleva la redefinición de los requisitos. Acordar con el cliente nuevos plazos de entrega.

Tabla 37: Medidas de prevención y contingencia Riesgo-010

Riesgo-011	
Descripción	Los miembros del equipo no se implican en el proyecto, y por lo tanto no alcanzan el nivel de rendimiento deseado.
Objetivo	Motivar a equipo de trabajo para que se involucren en el proyecto y aumenten su rendimiento.
Prevención	Establecer fórmulas de trabajo que mantengan al equipo motivado. Premiar a los trabajadores que alcancen el rendimiento exigido.
Contingencia	Advertir a los trabajadores de las consecuencias que implica su falta de interés en el proyecto. Si la actitud del equipo de desarrollo no cambia, sustituirlo por otro con mayor motivación e interés.

Tabla 38: Medidas de prevención y contingencia Riesgo-011

Riesgo-012	
Descripción	Los conflictos entre los miembros del equipo conducen a problemas en la comunicación y en el diseño, errores en la interfaz y tener que repetir algunos trabajos.
Objetivo	Mejorar el ambiente de trabajo y la armonía entre miembros del equipo con el fin de evitar los errores en los productos generados.
Prevención	Estudiar el ambiente de trabajo para saber el grado de motivación de los empleados. Organizar el equipo de tal manera que se puedan dividir las responsabilidades entre los miembros. Realizar dinámicas de grupo y actividades de ocio que ayude a establecer buenas relaciones entre los miembros del equipo de desarrollo.
Contingencia	Sustituir a los miembros conflictivos por otros empleados o bien redistribuir el trabajo entre el resto de integrantes.

Tabla 39: Medidas de prevención y contingencia Riesgo-012

Riesgo-013	
Descripción	Alguien de la plantilla abandona el proyecto antes de su finalización.
Objetivo	Evitar que la baja de un miembro del equipo repercuta en el desarrollo del proyecto con el fin de seguir cumpliendo los plazos y costes previstos.
Prevención	Ofrecer buenas condiciones laborales a los trabajadores para evitar que se desmotiven y cambien de empleo.
Contingencia	Repartir las tareas asignadas a ese miembro entre el resto de la plantilla. Contratar a alguien más si fuera necesario para mantener los plazos y presupuesto previstos.

Tabla 40: Medidas de prevención y contingencia Riesgo-013

Riesgo-014	
Descripción	Se producen pérdidas de información, bien en la documentación o en los productos desarrollados.
Objetivo	Recuperar la información perdida.
Prevención	Realizar copias de seguridad periódicas y almacenarlas en un lugar distinto al de trabajo (almacenamiento secundario). Utilizar sistemas de alimentación ininterrumpida.
Contingencia	Comprobar qué información se ha perdido y evaluar las consecuencias en cuanto a plazos de tiempo para repetir la información perdida. Extraer la información de las copias de seguridad realizadas.

Tabla 41: Medidas de prevención y contingencia Riesgo-014

3.5 Plan de pruebas

El propósito del plan de pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y manejo de riesgos de un proceso de pruebas. El plan de pruebas proporciona el marco dentro del cual el equipo de trabajo desarrolla las pruebas usando los recursos y la planificación dada. El objetivo que se persigue en esta sección es definir las pruebas que han de determinar la funcionalidad del sistema y asegurar la calidad del mismo, de forma que el sistema satisfaga los requisitos especificados por el cliente.

3.5.1 Definición de pruebas

El alcance de las pruebas que se describen en este plan es comprobar el correcto funcionamiento del sistema a desarrollar, para ello se describirán distintas pruebas para probar aspectos como la integridad de la base de datos, la funcionalidad del sistema, la interfaz de usuario y el rendimiento del sistema. Además de las pruebas que se detallan a continuación, se asume que durante el proceso de desarrollo se realizarán pruebas de carácter unitario que comprobarán cada uno de los componentes que forma parte del sistema de manera independiente.

Los tipos de pruebas que a realizar serían los siguientes:

- Pruebas de integridad de la base de datos
  - Verificar el acceso a la base de datos
  - Verificar el estado de la base de datos tras un proceso de actualización
- Pruebas del sistema
  - Verificar conexión/desconexión del sistema
  - Verificar proceso de consulta
  - Verificar la especificación del formato de archivo de entrada
  - Verificar la inserción de nuevos registros
  - Verificar el proceso de guardar resultados
  - Verificar el funcionamiento de la ayuda del sistema
  - Verificar la gestión de usuarios
- Pruebas de interfaz
  - Verificar la navegabilidad del sistema
- Pruebas de rendimiento
  - Verificar tiempo de respuesta de insertar nuevos registros
  - Verificar tiempo de respuesta de la ejecución de una consulta

En el caso de las pruebas de integridad de la base de datos, se probarán directamente sobre la base de datos, sin hacer uso intermedio de la interfaz. Para la verificación del acceso a la base de datos, se intentará acceder a la misma probando tanto con datos válidos como inválidos y se comprobará el resultado con la respuesta esperada. Para poder verificar que el estado de la base de datos tras un proceso de actualización es correcto, hay que inspeccionar la base de datos para comprobar que ha sido poblada como se esperaba. Estas pruebas son de tipo caja negra, ya que se ejecutan los procesos indicados y para posteriormente analizar los resultados generados. Se considerará que las pruebas han sido superadas si los procesos funcionan como fueron diseñados y sin corrupción de los datos.

Las pruebas del sistema se basan también en técnicas de caja negra, utilizando para ello la interfaz de usuario y analizando los resultados obtenidos. Estarán asociadas a casos de uso concretos, que permitirán conocer cuáles son las respuestas que se espera del sistema. Se probarán tanto con datos válidos como no válidos, para asegurar que la respuesta sea la esperada para todos y cada uno de los casos.

Las pruebas de interfaz permiten verificar la interacción del usuario con el software, su objetivo es asegurar que la interfaz de usuario provea al usuario el acceso apropiado para acceder y navegar por todas las funciones de la aplicación. Se comprobarán todas las ventanas de la aplicación y las características de navegabilidad, analizando los estados de los objetos para cada ventana.

La finalidad de las pruebas de rendimiento es medir los tiempos de respuesta, las tasas de transacción, así como otros factores sensibles al tiempo. Estas pruebas deben ser ejecutadas en repetidas ocasiones para poder estimar el rendimiento del sistema. También se podrá hacer uso de software especializado que permite analizar el rendimiento de un sistema de manera automática.

3.5.2 Especificación de pruebas

Los objetivos que se espera alcanzar tras la ejecución del plan de pruebas son comprobar que el sistema ofrece la funcionalidad requerida por el cliente y asegurar la calidad del mismo, esto se consigue demostrando que no existen errores ocultos en el sistema. Cada uno de los grupos de pruebas identificados en el apartado anterior tiene un objetivo distinto. En el caso de las pruebas de integridad de la base de datos, se trata de asegurar el correcto funcionamiento de la base de datos, en cuanto a su acceso como al estado de la misma tras realizar distintas operaciones. Las pruebas del sistema aseguran que el sistema funcione tal y como se especifique en los casos de uso, y que por tanto satisfaga las necesidades del cliente. Las pruebas de interfaz comprueban que el sistema ofrezca una navegación intuitiva y correcta. Finalmente, las pruebas de rendimiento comprueban que los tiempos de respuesta del sistema no sean excesivos y se ajusten a las restricciones impuestas.

Para la ejecución de estas pruebas es necesario contar con una serie de recursos, tanto humanos como de sistema. El equipo de pruebas estará formado por tres personas: un jefe de pruebas y dos probadores. El papel del jefe de pruebas lo desempeñará el analista del sistema, mientras que los probadores serán los dos programadores contratados para el desarrollo del proyecto. Los recursos de sistema necesarios para llevar a cabo el plan de pruebas son los siguientes:

- **Un servidor:** con Windows XP, procesador a 2 GHz, memoria RAM de 3 GB y disco duro de 100 GB.
- **Dos ordenadores cliente:** con Windows XP, procesador a 1,5 GHz, memoria RAM de 1GB y disco duro de 80 GB.

El plan de pruebas se ejecutará una vez finalizada la implementación del sistema y antes de entregárselo al cliente. En caso de que alguna prueba no sea superada, será necesario arreglar el sistema y volver a pasar todas las pruebas para comprobar que el resto de funcionalidad sigue siendo correcta. Para este proceso, se calcula dedicarle un tiempo de una semana, sin embargo, es difícil de saber con precisión el tiempo que puede llevar arreglar los problemas que se encuentren.

Solución

4 Solución

Este capítulo describe la solución adoptada para resolver el problema planteado en el capítulo de estudio del problema. El capítulo se centrará en el proceso de desarrollo seguido para alcanzar la solución, describiendo con detalle todos los pasos seguidos y las decisiones tomadas durante el proceso.

4.1 Descripción de la solución

La solución adoptada corresponde con el desarrollo de una aplicación web que tiene como misión principal simplificar el proceso de actualización de la base de datos bibliográfica del LEMI, de forma que este proceso pueda ser realizado por cualquier usuario autorizado de forma sencilla e intuitiva. Así, la información podrá ser actualizada con mayor frecuencia que como hoy en día. Además, la aplicación proporcionará información que permitirá al usuario conocer el resultado del proceso, informándosele de los archivos nuevos que se han añadido a la base de datos, los que han sido modificados y cuáles han generado algún tipo de incidencia. La aplicación también ofrecerá dos métodos para consultar la información contenida en la base de datos. La primera de ellas será a través de sencillos formularios proporcionados por la aplicación, la otra es a través de sentencias SQL que permitirán ejecutar consultas más complicadas. En todo caso, la información recuperada de la base de datos se mostrará en forma de tablas y podrá ser almacenada en distintos formatos para su posterior estudio y recuperación.

4.2 El proceso de desarrollo

Para el desarrollo del proyecto se ha seguido un ciclo de vida en cascada. Se define como ciclo de vida del software a las diversas etapas por las que debe pasar un sistema, desde la formulación del problema, seguido por el análisis y especificación de requisitos, diseño, implementación, integración y pruebas, para terminar con la fase operacional donde se mantiene y extiende el sistema [5]. Se ha elegido el modelo en cascada debido a que partimos con unos requisitos bien definidos, establecidos por las necesidades del cliente y que no van a cambiar a lo largo del desarrollo del proceso. Además, es un modelo que permite desarrollar el software de manera rápida y sencilla y que asegura la finalización de una etapa antes de continuar con la siguiente, agilizando el proceso. Las fases que se han seguido en este proceso pueden verse en la Figura 9.

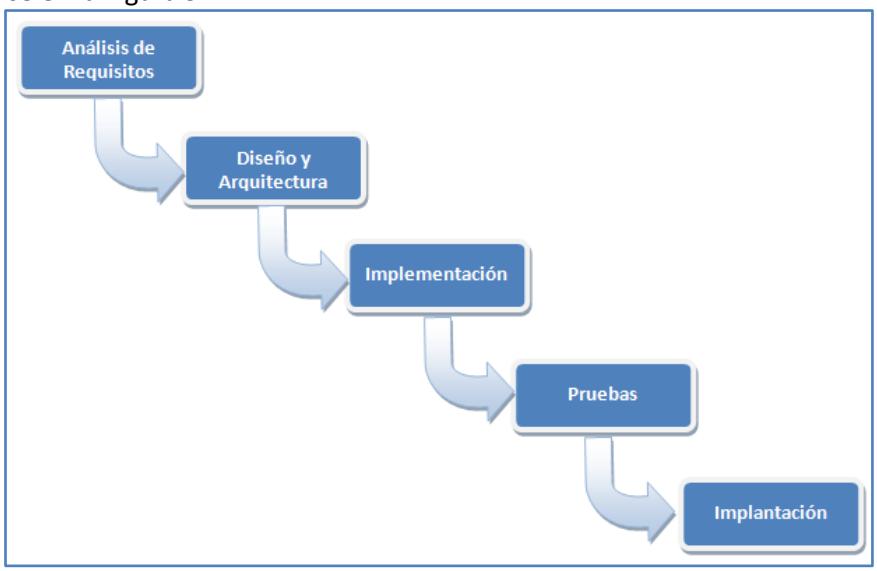

Figura 9: Proceso de desarrollo siguiendo el modelo en cascada

A continuación se describen las fases que se han seguido durante el desarrollo de la aplicación y las cuales se tratarán extensamente a lo largo de este capítulo:

- **Análisis de requisitos:** se trata de la primera etapa para la creación de un producto software. Presenta ciertas dificultades ya que aunque el cliente sabe qué es lo que quiere que haga la aplicación, a veces estos requisitos son incompletos, ambiguos y contradictorios, por lo que se requiere de suficiente habilidad para especificarlos de manera correcta y evitar problemas en fases posteriores.
- **Diseño y arquitectura:** en esta fase se determina como funcionará el sistema de manera general, incorporando consideraciones de la implementación tecnológica.
- **Implementación:** consiste en traducir el diseño del sistema a código, representa la fase de programación.
- **Pruebas:** consiste en comprobar que el sistema realice correctamente las tareas indicadas en la especificación de requisitos, se seguirá el plan de pruebas establecido en la sección 5.1.1.
- **Implantación:** en esta fase se integra el sistema con la organización que lo solicita, en este caso el LEMI y se comprueba que interactúe correctamente con los usuarios que lo utilizan.

4.2.1 Análisis

En esta fase se pretende determinar cuáles son las necesidades del cliente que debe cubrir el sistema, para ello es necesario establecer una serie de reuniones y entrevistas con el usuario final o cliente. Mediante estas reuniones se identifican las metas globales, se analizan las perspectivas del cliente, sus necesidades y requerimientos, así como todos aquellos puntos que puedan ayudar a la identificación y desarrollo del proyecto. Esta fase se puede dividir en dos tareas principales:

- **Definición de requisitos:** consiste en terminar los requisitos generales del sistema mediante sesiones de trabajo con los usuarios.
- **Especificación de requisitos:** se hará una descripción completa del comportamiento del sistema que se va a desarrollar a partir de los requisitos previamente identificados.

Definición de requisitos

La definición de requisitos trata de analizar y determinar cuáles son los objetivos y necesidades del cliente, dando como resultado un conjunto de requisitos que el sistema debe satisfacer. Será necesario establecer varias sesiones de trabajo con el cliente con el fin de obtener un conjunto de requisitos lo más completo y estable posible para, de esta manera, evitar problemas en otras fases que conduzcan a tener que rehacer parte del trabajo.

Los requisitos identificados pertenecen a uno de estos dos grupos, funcionales o no funcionales. Los requisitos funcionales como su nombre indica expresan una funcionalidad del sistema, es decir, la capacidad de acción del mismo, qué es lo que tiene que hacer. Los requisitos no funcionales, en cambio, expresan una característica requerida del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo que supone una restricción al mismo. Los requisitos no funcionales a su vez se dividen en varios tipos:

- **Rendimiento:** son aquellos relacionados con la capacidad, velocidad y fiabilidad del sistema.

- **Operativos:** recogen las características del entorno físico y técnico en el que se va a operar.
- **Seguridad:** están relacionados con la seguridad de la aplicación y la información en ella almacenada, especificando quiénes y en qué casos están autorizados a acceder a la aplicación.
- **Normativos:** aquellos relacionados con factores políticos, legales y/o culturales que afectan al sistema.

Cada requisito identificado se ha definido en una tabla con los siguientes campos:

- **Identificador:** código para identificar de manera única cada requisito siguiendo la nomenclatura: <R><grupo><tipo>_<número>. Donde el <grupo> identifica si se trata de un requisito funcional (F) o no funcional (NF) y el <tipo> identifica si el requisito no funcional es de rendimiento (R), operativo (O), de seguridad (S) o normativo (N). El número estará formado por tres cifras empezando por 001.
- **Descripción:** breve descripción del requisito.
- **Dependencias:** identifica a otros requisitos con los cuales el requisito descrito mantiene algún tipo de relación.
- **Claridad:** establece si el requisito admite una y sólo una interpretación.
- Fuente: define el origen del requisito.
- **Prioridad:** mide la prioridad del requisito con el propósito de definir una estrategia de desarrollo. La prioridad puede tomar valores entre alta, media o baja.
- **Necesidad:** este valor establece cómo de importante es para el cliente que el requisito sea tenido en cuenta. Puede tomar los valores de esencial, deseable y opcional.
- **Estabilidad:** este valor trata de estimar si el requisito es vulnerable a sufrir cambios en el futuro. Los valores que puede tomar son: alta, media o baja.
- **Verificabilidad:** define si el requisito es verificable o no.

RF-001			
Descripción	Los usuarios se identifican mediante un identificador y una contraseña, además se guardan otros datos de los mismos: nombre, apellidos y perfil.		
Dependencias	-		
Claridad	Sí	**Fuente**	Analista
Prioridad	Media	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 42: Definición del requisito RF-001

RF-002			
Descripción	El acceso al sistema estará restringido a usuarios registrados, por lo que para acceder al mismo el usuario deberá validarse mediante su identificador y contraseña.		
Dependencias	RF-001		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 43: Definición del requisito RF-002

RF-003			
Descripción	Sólo los usuarios con cargo de administrador podrán añadir a otros usuarios al sistema.		
Dependencias	RF-002		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 44: Definición del requisito RF-003

RF-004			
Descripción	Sólo los usuarios con cargo de administrador podrán modificar a otros usuarios del sistema.		
Dependencias	RF-002		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Media	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 45: Definición del requisito RF-004

RF-005			
Descripción	Sólo los usuarios con cargo de administrador podrán eliminar a otros usuarios del sistema.		
Dependencias	RF-002		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 46: Definición del requisito RF-005

RF-006			
Descripción	El sistema permitirá a los usuarios conectados, desconectarse en el momento que lo deseen.		
Dependencias	RF-002		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 47: Definición del requisito RF-006

RF-007			
Descripción	El sistema permitirá hacer consultas a la base de datos mediante una serie de formularios.		
Dependencias	RF-002		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 48: Definición del requisito RF-007

RF-008	
Descripción	El sistema permitirá hacer consultas a la base de datos introduciendo sentencias en lenguaje SQL.
Dependencias	RF-002

Claridad	Sí	**Fuente**	Analista
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 49: Definición del requisito RF-008

RF-009			
Descripción	El resultado de cualquier tipo de consulta se mostrará en forma de tabla.		
Dependencias	RF-007, RF-008		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 50: Definición del requisito RF-009

RF-010			
Descripción	El sistema permitirá al usuario determinar el formato que tendrán los ficheros de entrada con los registros a insertar.		
Dependencias	RF-002		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 51: Definición del requisito RF-010

RF-011			
Descripción	El sistema permitirá insertar nuevos registros a partir de ficheros con un formato previamente determinado para actualizar la base de datos.		
Dependencias	RF-002, R-010		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 52: Definición del requisito RF-011

RF-012			
Descripción	El sistema proporcionará información sobre el resultado de la operación de inserción, indicando el número de registros añadidos, modificados y que han originado algún tipo de problema.		
Dependencias	RF-011		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 53: Definición del requisito RF-012

RF-013			
Descripción	El usuario podrá ver cuáles son los registros añadidos, modificados y con incidencias en forma de tabla.		
Dependencias	RF-012		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 54: Definición del requisito RF-013

RF-014			
Descripción	El sistema permitirá guardar la información mostrada en forma de tablas por la aplicación en formato pdf y csv.		
Dependencias	RF-009, RF-013		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 55: Definición del requisito RF-014

RF-015			
Descripción	El sistema proporcionará un mecanismo de ayuda orientando al usuario en la utilización de la aplicación.		
Dependencias	RF-002		
Claridad	Sí	**Fuente**	Analista
Prioridad	Baja	**Necesidad**	Opcional
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 56: Definición del requisito RF-015

RNFR-001			
Descripción	El acceso a la aplicación debería hacerse en menos de 5 segundos el 80% de las veces.		
Dependencias	-		
Claridad	Sí	**Fuente**	Analista
Prioridad	Media	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 57: Definición del requisito RNFR-001

RNFO-001			
Descripción	El sistema y la base de datos estarán instalados en un servidor.		
Dependencias	-		
Claridad	Sí	**Fuente**	Analista
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 58: Definición del requisito RNFO-001

RNFO-002			
Descripción	El sistema requiere de un navegador web para ser accedido desde el lado del cliente.		
Dependencias	RNFO-001		
Claridad	Sí	**Fuente**	Analista
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 59: Definición del requisito RNFO-002

RNFO-003	
Descripción	La aplicación será compatible con los siguientes navegadores: Mozilla Firefox e Internet Explorer.

Dependencias	RNFO-002		
Claridad	Sí	**Fuente**	Analista
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 60: Definición del requisito RNFO-003

RNFO-004			
Descripción	El idioma de la aplicación será el castellano.		
Dependencias	-		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 61: Definición del requisito RNFO-004

RNFS-001			
Descripción	El sistema garantizará la confidencialidad, integridad y disponibilidad de los datos.		
Dependencias	-		
Claridad	Sí	**Fuente**	Analista
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 62: Definición del requisito RNFS-001

RNFS-002			
Descripción	El sistema requiere de autenticación por parte de los usuarios para poder ser accedido.		
Dependencias	RF-001, RF-002		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 63: Definición del requisito RNFS-002

RNFS-003			
Descripción	Habrá distintos perfiles de usuario que determinarán las operaciones que pueden realizar.		
Dependencias	RF-001		
Claridad	Sí	**Fuente**	Cliente
Prioridad	Alta	**Necesidad**	Esencial
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 64: Definición del requisito RNFS-003

RNFN-001			
Descripción	Los datos de los usuarios del sistema deberán cumplir la LOPD [14].		
Dependencias	-		
Claridad	Sí	**Fuente**	Analista
Prioridad	Alta	**Necesidad**	Deseable
Estabilidad	Alta	**Verificabilidad**	Sí

Tabla 65: Definición del requisito RNFN-001

Especificación de requisitos

Una vez definidos los requisitos que debe cumplir nuestro sistema es necesario dar un paso más para definir la estructura, la funcionalidad y comportamiento que tendrá el sistema. Para ello, en primer lugar definiremos los distintos casos de uso en que puede dividirse el sistema. Un caso de uso se define como la descripción de una secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica , es decir, documentan el comportamiento del sistema desde el punto de vista del usuario, representando las funciones que puede ejecutar [4]. La especificación de los requisitos mediante casos de uso ayuda a asegurar que el sistema se desarrolla correctamente, proporcionando una buena base para la verificación y validación, así como un excelente vehículo de comunicación con el usuario. A continuación se describen los distintos casos de uso definidos para el sistema:

CU-001: Conectarse al sistema		
Descripción	Caso de uso que describe los pasos necesarios para que un usuario pueda conectarse al sistema para poder acceder a toda su funcionalidad.	
Precondiciones	Conocer la dirección URL para acceder a la aplicación. El usuario debe estar registrado en el sistema.	
Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	Escribe la dirección URL de la aplicación.	El sistema muestra la interfaz de bienvenida, para que el usuario se conecte.
2	El usuario introduce su identificador y contraseña y pulsa el botón acceder.	El sistema comprueba que los datos introducidos coinciden con los de algún usuario registrado. En caso afirmativo muestra la pantalla de bienvenida para usuarios conectados.
Post-condiciones	El usuario está ya conectado y puede acceder a todos los servicios para los que esté autorizado.	
Flujo alternativo		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El usuario no está registrado o introduce un identificador y/o contraseña erróneos.	El sistema comprueba que los datos introducidos coinciden con los de algún usuario registrado, como no es el caso vuelve a mostrar la interfaz de conexión y el mensaje de error de que los datos introducidos no son correctos.

Tabla 66: Descripción caso de uso CU-001

CU-002: Añadir usuario al sistema		
Descripción	Caso de uso que describe cómo añadir un nuevo usuario al sistema por parte de un usuario administrador.	
Precondiciones	El usuario debe estar conectado al sistema. El usuario debe tener el perfil de administrador.	
Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El administrador selecciona la opción	El sistema muestra una interfaz para

	de añadir nuevo usuario al sistema.	introducir los datos del usuario.
2	El administrador introduce el identificador, contraseña, nombre, apellidos y perfil del usuario en el formulario y pulsa aceptar.	El sistema comprueba que no exista ningún otro usuario con el mismo identificador y procede a su inserción. Le muestra un mensaje con el resultado de la operación.
Post-condiciones	El nuevo usuario está registrado en la aplicación y puede conectarse al sistema con su identificador y contraseña.	
Flujo alternativo		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El administrador introduce un identificador que ya está registrado para otro usuario.	El sistema muestra un mensaje solicitando al administrador que elija otro identificador en la interfaz de añadir usuario.

Tabla 67: Descripción caso de uso CU-002

CU-003: Modificar un usuario del sistema		
Descripción	Caso de uso que describe cómo modificar los datos de algún usuario registrado en el sistema por parte de un administrador.	
Precondiciones	El usuario debe estar conectado al sistema. El usuario debe tener el perfil de administrador.	
Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El administrador selecciona la opción de modificar usuario del sistema.	El sistema muestra una interfaz con los distintos usuarios registrados.
2	El administrador selecciona de la interfaz el usuario que quiere modificar los datos.	El sistema muestra una interfaz con los datos del usuario elegido para que puedan ser modificados.
3	El administrador modifica los campos que considere convenientes y pulsa aceptar.	El sistema guarda los campos en los datos del usuario modificado y muestra una interfaz con el resultado de la operación.
Post-condiciones	-	

Tabla 68: Descripción caso de uso CU-003

CU-004: Eliminar un usuario del sistema		
Descripción	Caso de uso que describe cómo eliminar algún usuario registrado en el sistema por parte de un administrador.	
Precondiciones	El usuario debe estar conectado al sistema. El usuario debe tener el perfil de administrador. El usuario a eliminar debe estar registrado en el sistema.	
Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El administrador selecciona la opción de eliminar usuario del sistema.	El sistema muestra una interfaz con los distintos usuarios registrados.
2	El administrador selecciona de la interfaz el usuario que quiere eliminar.	El sistema requiere confirmación de operación por parte del administrador.
3	El administrador confirma su decisión de eliminar el usuario.	El usuario es borrado de la base de datos del sistema.

Post-condiciones	El usuario es borrado del sistema y no podrá acceder a la aplicación.

Tabla 69: Descripción caso de uso CU-004

CU-005: Consultar la base de datos		
Descripción	Caso de uso que describe cómo efectuar una consulta a la base de datos.	
Precondiciones	El usuario debe estar conectado al sistema.	
Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El usuario selecciona una de las opciones de consulta que ofrece la aplicación: consulta SQL o consulta por opciones de búsqueda.	El sistema muestra la interfaz correspondiente a la opción de consulta elegida.
2	El usuario rellena los formularios de consulta con las opciones adecuadas.	El sistema muestra en una tabla todos los registros que coinciden con la consulta elegida.
Post-condiciones	-	

Tabla 70: Descripción caso de uso CU-005

CU-006: Determinar formato de entrada		
Descripción	Caso de uso que describe los pasos a seguir para establecer el formato de entrada que tiene el archivo con los registros para poder ser leído por la aplicación y determinar cuáles son cada uno de los campos.	
Precondiciones	El usuario debe estar conectado al sistema.	
Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El usuario selecciona la opción de determinar formato de entrada de entre las que ofrece el sistema.	El sistema muestra la interfaz con los formularios que el usuario deberá rellenar para establecer un nuevo formato de entrada.
2	El usuario elige un nombre para identificar ese formato de archivo, la tabla donde se insertarán los datos de ese archivo y los identificadores que separaran cada uno de los campos.	El sistema comprueba que no exista ya ese identificador de formato de archivo y procede a almacenar la información. Muestra al usuario una interfaz con el resultado de la operación.
Post-condiciones	El nuevo formato estará disponible para ser elegido cuando el usuario quiera añadir nuevos registros contenidos en un fichero a la base de datos.	
Flujo alternativo		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El usuario elige un identificador de formato que ya existe.	El sistema muestra un mensaje solicitando al usuario que elija otro identificador en la interfaz de determinar formato de entrada.

Tabla 71: Descripción caso de uso CU-006

CU-007: Insertar registros	
Descripción	En este caso de uso se describen los pasos que se deben tomar para añadir nuevos registros o actualizar los ya existentes en la base de datos.

Precondiciones	El usuario debe estar conectado al sistema. El fichero de entrada con los registros debe tener un formato que coincida con uno previamente especificado.

Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El usuario selecciona de entre las opciones que ofrece el sistema la de insertar registros.	El sistema muestra la interfaz correspondiente a la opción de insertar registros.
2	El usuario selecciona el formato del registro y la localización en su sistema de ficheros del archivo que contiene los registros a añadir.	El sistema procede a insertar/modificar los registros pertinentes y muestra el resultado de la operación indicando el número de registros añadidos, modificados y con incidencia.
3	El usuario selecciona una de las opciones: archivos modificados, añadidos o con incidencias, para ver cuáles hay en cada grupo.	El sistema muestra una tabla con los registros pertenecientes al grupo seleccionado por el usuario.

Post-condiciones	La base de datos del sistema se actualiza con la información de los nuevos registros indicados por el usuario.

Flujo alternativo		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El fichero tiene un formato incorrecto que no corresponde con el seleccionado.	El sistema muestra un mensaje de error indicando que el fichero no se ajusta al formato seleccionado.

Tabla 72: Descripción caso de uso CU-007

CU-008: Guardar resultados	
Descripción	Caso de uso que describe cómo guardar los resultados mostrados en forma de tabla por parte de la aplicación en un fichero para su posterior consulta o manipulación.
Precondiciones	El usuario debe estar conectado al sistema. El sistema debe estar en modo “mostrar resultados en forma de tabla” tras haber realizado el usuario una consulta o una inserción.

Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El usuario selecciona la opción de guardar resultado en archivo.	El sistema muestra una interfaz donde le pide al usuario que seleccione el tipo de formato en que quiere guardar la tabla, el nombre y la ubicación en su sistema de ficheros.
2	El usuario rellena los campos solicitados por el sistema.	El sistema procede a guardar los ficheros mostrados como resultado en un fichero en el formato indicado por el usuario.

Post-condiciones	-

Tabla 73: Descripción caso de uso CU-008

CU-009: Consultar ayuda	
Descripción	Caso de uso que describe qué pasos debe seguir el usuario para acceder a la ayuda del sistema.

Precondiciones	El usuario debe estar conectado al sistema.	
Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El usuario selecciona la opción de consultar ayuda.	El sistema muestra una interfaz con un manual sobre cómo utilizar la aplicación.
Post-condiciones	-	

Tabla 74: Descripción caso de uso CU-009

CU-010: Desconectarse		
Descripción	Este caso de uso describe los pasos que debe seguir un usuario para desconectarse del sistema.	
Precondiciones	El usuario debe estar conectado al sistema.	
Flujo básico		
Línea	**Acción del actor**	**Respuesta del sistema**
1	El usuario pulsa sobre la opción de desconectar que ofrece el sistema.	El sistema desconecta al usuario del mismo y muestra la interfaz de bienvenida para usuarios no conectados.
Post-condiciones	El usuario está desconectado y mientras no vuelva a conectarse al sistema no podrá acceder a las operaciones para las que está autorizado.	

Tabla 75: Descripción caso de uso CU-010

Tras describir en detalle los distintos casos de uso de los que se compone nuestro sistema, es conveniente recogerlos todos ellos en un diagrama de casos de uso mediante UML 2.0 (ver Figura 10), ya que de esta forma se puede tener una visión global de todos ellos, así como de los actores que interaccionan con el sistema. Nuestro sistema cuenta con dos tipos de actores (usuarios del sistema):

- **Administrador:** usuario con privilegios que incluyen no sólo la realización de operaciones comunes sobre la aplicación, sino también la gestión de usuarios.
- **Usuario normal:** usuario con privilegios restringidos, puede hacer uso de todas las operaciones propias de la aplicación, pero no gestionar usuarios.

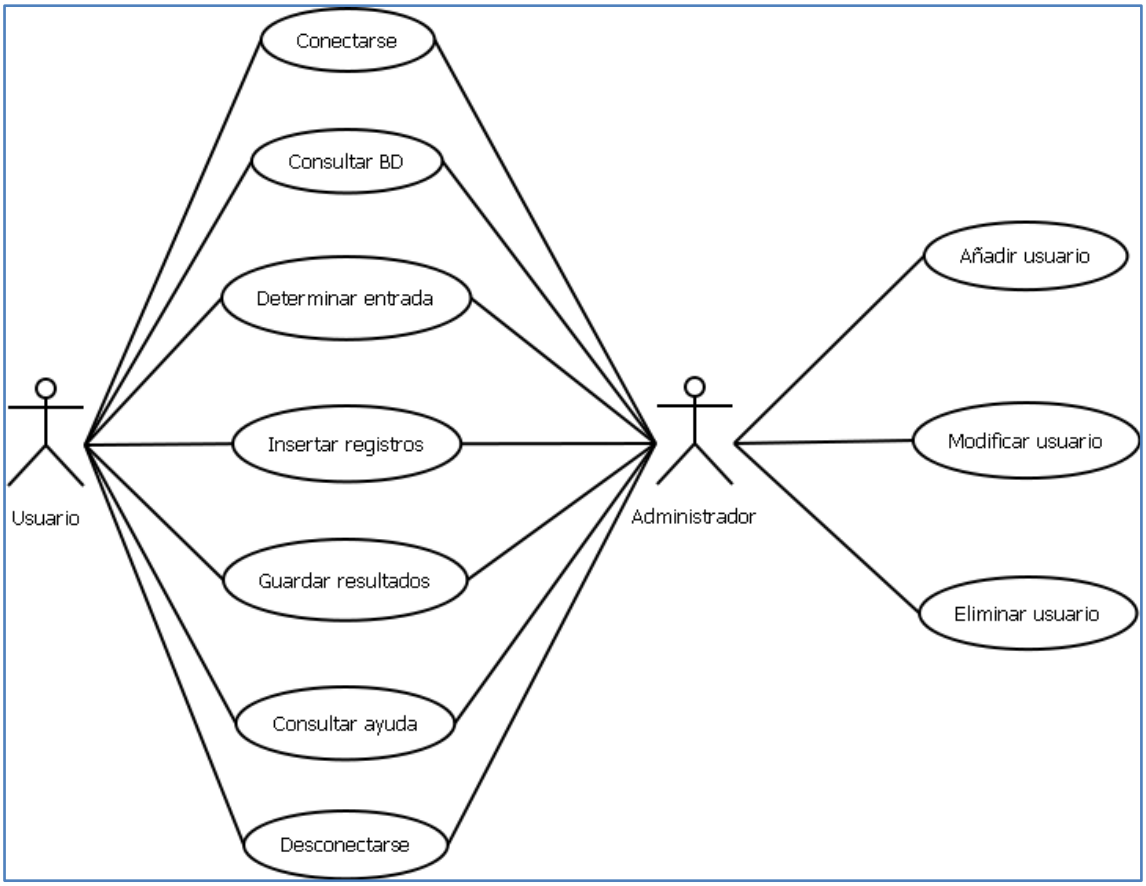

Figura 10: Diagrama de casos de uso

Después de determinar los actores que forman parte del sistema y las acciones que pueden llevar a cabo, es necesario descubrir qué objetos intervienen en los procesos y operaciones. El diagrama de secuencia es uno de los más efectivos para modelar la interacción entre objetos de un sistema, ya que muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. Mientras que el diagrama de casos de uso nos ha aportado una vista general del sistema y sus operaciones, los diagramas de secuencia contienen detalles de implementación del escenario, incluyendo los objetos y clases, y los mensajes intercambiados por los objetos. A continuación se muestran los diagramas de secuencia correspondientes al proceso de inserción de registros (Figura 11) y al proceso de consulta mediante formularios (Figura 12). En el diagrama del proceso de inserción, muestra también la secuencia de mensajes que es el resultado de la acción de usuario solicitando ver los registros que han sido modificados. Por otro lado, el proceso de guardar los registros como resultado de una consulta o un proceso de inserción puede verse en el diagrama secuencial de la consulta.

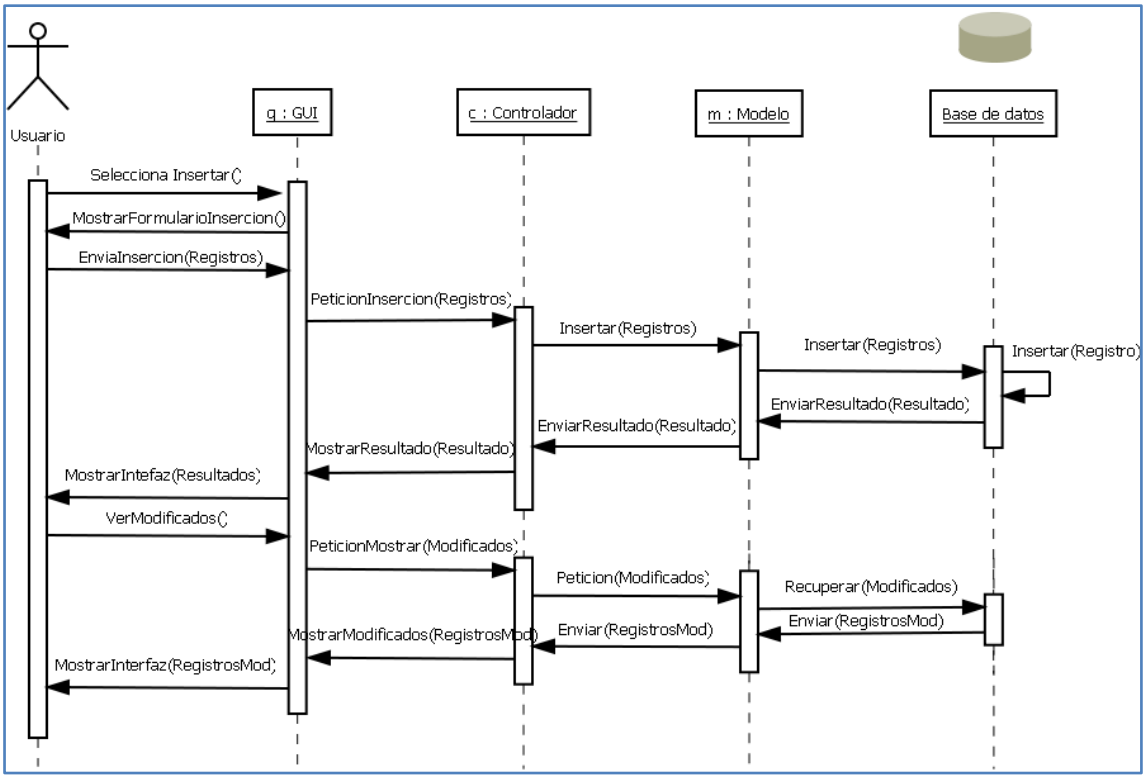

Figura 11: Diagrama secuencial del proceso de inserción

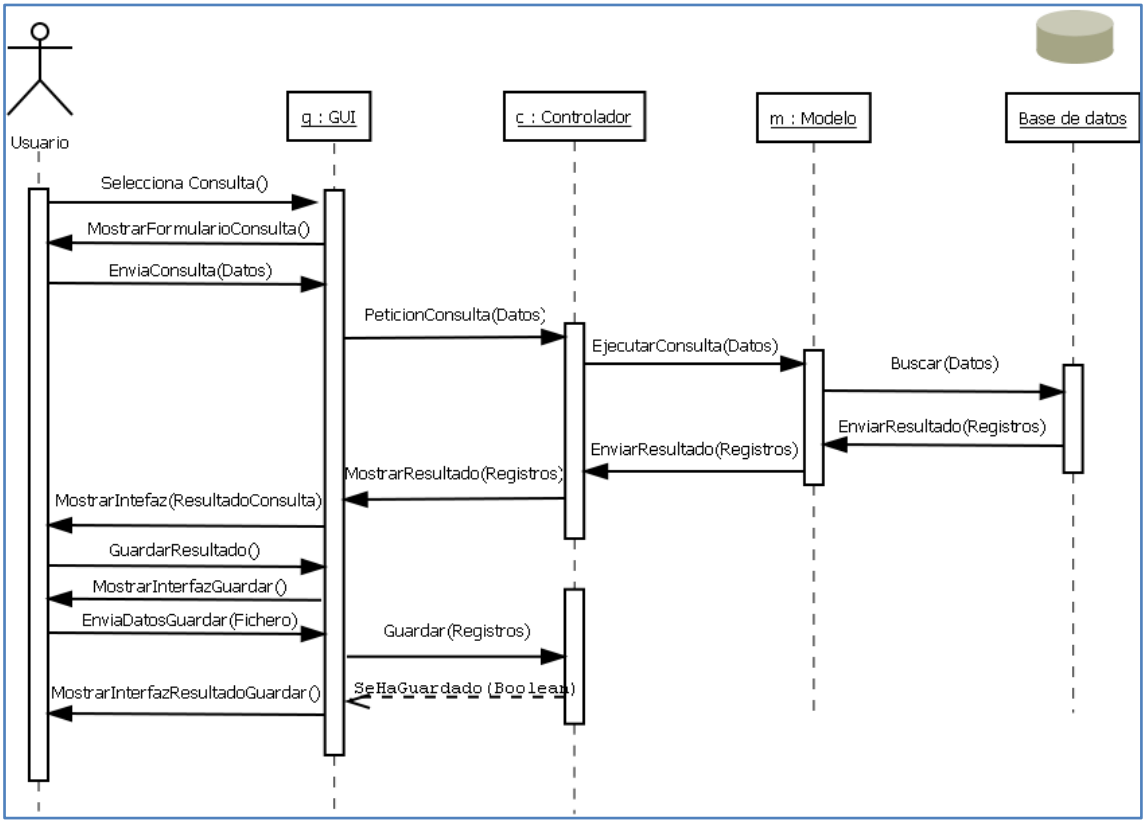

Figura 12: Diagrama secuencial del proceso de consulta

4.2.2 Diseño

Concluida la primera fase de análisis del sistema, es necesario sentar las bases que van a permitir construirlo, esto se hace en la fase de diseño. El diseño consiste en un proceso iterativo a través del cual se traducen los requisitos identificados en la fase de análisis en una

representación del software con detalles suficientes como para permitir su realización física. Esta fase puede dividirse en dos etapas, diseño del sistema y diseño detallado. En el diseño del sistema se define la arquitectura del sistema y se especifica el entorno tecnológico elegido. El diseño detallado es un diseño de bajo nivel, donde se describen las capas de persistencia, modelo e interfaz del sistema. En las siguientes secciones se describen tanto el diseño del sistema como el detallado que se han realizado para este proyecto.

Diseño de sistema

En el diseño del sistema se describen tanto la arquitectura como las tecnologías que se va a utilizar durante el proceso de desarrollo del sistema.

Arquitectura del sistema

El diseño de la arquitectura es el diseño de más alto nivel de la estructura de un sistema; consiste en un conjunto de abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software de un sistema de información, cubriendo todas las necesidades especificadas por el cliente y recogidas por el analista en los requisitos. La elección de una u otra arquitectura es un factor determinante a la hora de escoger la tecnología que vamos a usar en nuestro sistema, ya que no todas las tecnologías son aptas para todas las arquitecturas.

Generalmente, no es necesario diseñar una arquitectura por cada sistema a diseñar, sino que lo normal es adoptar una arquitectura conocida y evaluarla en función de las características del sistema. Existen diversas arquitecturas muy conocidas como la monolítica, cliente-servidor, de tres capas, orientada a servicios, etc. Debido a las características de este proyecto la arquitectura elegida es una arquitectura en tres capas, correspondientes a la presentación, el negocio y los datos. La principal ventaja de esta arquitectura es que el código está separado en capas distintas según su funcionalidad y facilita cualquier modificación que pueda surgir en el futuro. Además, permite la programación de cada capa de manera independiente, simplificando el proceso de desarrollo del sistema. A continuación se describen cada una de las tres capas de esta arquitectura:

- **Capa de presentación:** es la que ve el usuario, al que se le presenta el sistema, se le comunica la información y desde donde se captura su información.
- **Capa de negocio:** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso.
- **Capa de datos:** es donde residen los datos y la encargada de acceder a los mismos. Está formada por el gestor de bases de datos, donde se realiza el almacenamiento de los mismos y se reciben las solicitudes de almacenamiento o recuperación de la información desde la capa de negocio.

Para desarrollar esta arquitectura se va a usar el patrón Modelo-Vista-Controlador (MVC), que también distribuye los componentes del sistema en tres partes: modelo, vista y controlador. No se trata de una implementación pura del modelo de tres capas, ya que en el modelo de tres capas la capa de presentación y la de datos no se comunican directamente, sólo a través de la capa de negocio. En el patrón MVC sí que existe una comunicación directa entre las tres capas, como puede verse en la Figura 13.

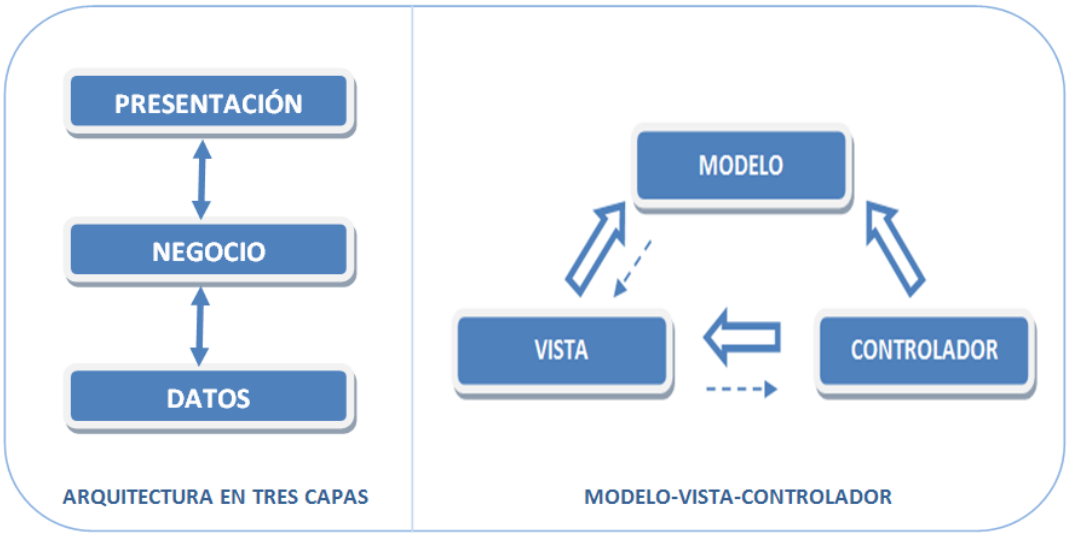

Figura 13: Comparativa arquitectura en tres capas con patrón arquitectónico MVC

El sistema a desarrollar por tanto ha de tener la funcionalidad distribuida en las capas en que se divide el patrón MVC, y que se describen a continuación:

- **Modelo:** representación de la información que maneja la aplicación. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista:** representación del modelo en forma gráfica, disponible para la interacción con el usuario. En el caso de las aplicaciones web, la vista es la página HTML sobre la que el usuario puede realizar operaciones. En esta capa se recogen todas las clases necesarias para generar la interfaz que verá el usuario.
- **Controlador:** es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso de ser necesario.

Para entender mejor cómo se va a adaptar el sistema a desarrollar a la arquitectura MVC, es necesario explicar las tecnologías que se van a usar durante el proceso de desarrollo, ya que como se ha explicado anteriormente no todas las tecnologías se pueden adaptar a cualquier arquitectura.

Descripción de tecnologías

En primer lugar es conveniente reseñar que el lenguaje de programación elegido para el desarrollo de la aplicación es Java, en concreto la plataforma Java EE [13], lo que limitará el uso de tecnologías a aquellas relacionadas con este lenguaje de programación. Además, para el desarrollo de la solución se ha empleado un framework elaborado por el grupo de investigación DEI, perteneciente a la Universidad Carlos III. Este framework hace uso de las tecnologías escogidas para el desarrollo de la aplicación, además de proporcionar una serie de elementos básicos que facilitan el proceso de implementación de la aplicación. Por otro lado, el framework ofrece también una ordenación de las distintas partes de la aplicación, proporcionando librerías de componentes previamente desarrollados para la capa de la vista así como una serie de servicios para la comunicación entre las distintas capas.

Como ya se ha explicado previamente se va a seguir el patrón de arquitectura MVC durante el desarrollo de la aplicación, por lo tanto a continuación se van a explicar las tecnologías que se van utilizar para implementar cada una de las capas del patrón MVC.

Vista

Esta capa se encarga de mostrar al usuario la interfaz con la interactuará, por lo tanto esta interfaz debe poder ser interpretada por los principales navegadores del mercado. La tecnología elegida para implementar y generar la interfaz es el framework de desarrollo JSF [31], ya que está basado en el patrón MVC y permite el desarrollo de componentes en Java. La vista está representada por componentes de interfaz de usuario (UI Components) que son elementos individuales que representan el estado y comportamiento de una amplia variedad de elementos funcionales de interfaz de usuario, desde los más básicos hasta otros de carácter más complejo. Además, para facilitar el desarrollo de estos componentes se hará uso de RichFaces [25], que es una librería de componentes para JSF y además permite la integración de AJAX en nuestra aplicación.

Controlador

La misión de esta capa es gestionar y responder las solicitudes del usuario, para ello ha de procesar la información necesaria y hacer las modificaciones pertinentes del modelo. Para la implementación del controlador se va a utilizar otra de las características que ofrece el framework JSF, en este caso el FacesServlet, responsable de procesar y usar los metadatos almacenados para gestionar la navegación y el ciclo de vida del modelo. Recibe eventos de la intefaz de usuario y los relaciona con la lógica de negocio apropiada. Se relaciona muy estrechamente con los managed beans, ya que son los metadatos de los que depende el FacesServlet, encapsulando las interfaces de la lógica de negocio. Los managed beans son instanciados en tiempo de ejecución y se destruyen cuando dejan de ser útiles. De ellos también depende la navegación del sistema.

Modelo

Es la capa encargada de gestionar y asegurar la integridad de la información y los datos que usa la aplicación. Para el desarrollo de esta capa se va a hacer uso de EJB 3.0 [8], una arquitectura que define la forma de construir componentes distribuidos del lado del servidor. Esta tecnología garantiza que los componentes programados sean escalables, eficaces y seguros, además de sencillos de desarrollar. Estos componentes se dividen en tres grupos: beans de sesión, de entidad y dirigidos por mensajes. En nuestro caso, para el desarrollo del sistema nos centraremos en los EJB de sesión, que se encargan de resolver la lógica de negocio de la aplicación. Los beans de sesión se pueden dividir a su vez en dos tipos dependiendo de si se guarda o no el estado.

Para facilitar el acceso a los datos, se hará uso del patrón DAO (Data Access Object), ya que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento. El motivo de que nos ha llevado a elegir este patrón es que cualquier objeto de negocio no requiere tener conocimiento directo del destino final de la información que manipula. De esta forma, se aísla a la aplicación de la tecnología de persistencia elegida, en nuestro caso EJB, facilitando el proceso de actualización sin afectar a otras partes de la aplicación.

Finalmente, y para tener una visión global de las tecnologías que se van a emplear para el desarrollo de cada una de las capas asociadas al patrón MVC, estas se recogen en la Figura 14.

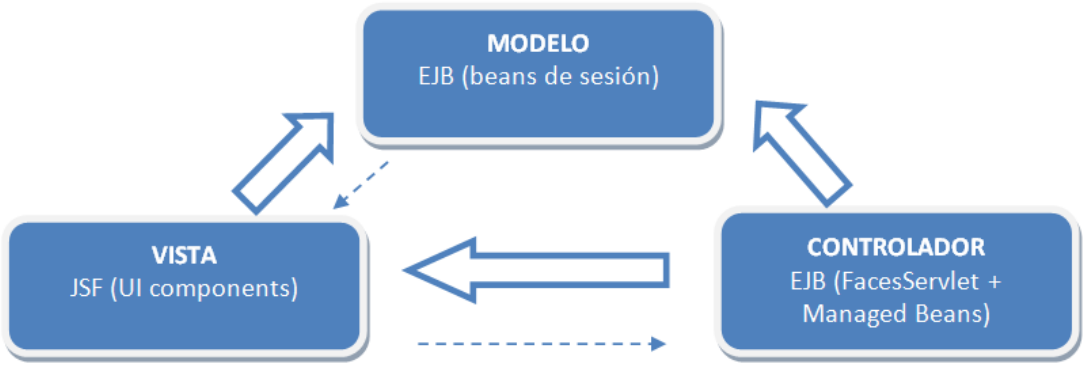

Figura 14: Tecnologías a aplicar para el MVC

Diseño detallado

En esta sección damos un paso más en el diseño de la aplicación, describiendo con más detalle cómo deberá desarrollarse la aplicación y cómo se van a aplicar las distintas tecnologías elegidas en las capas de persistencia, modelo e interfaz.

Persistencia

En esta sección se va a explicar con más detalle la capa de persistencia, es decir, la capa donde los datos de la aplicación están almacenados. Para guardar los datos de la aplicación se va a usar el SGBD MySQL. El modelo de datos que corresponde a la capa de persistencia, puede verse en la Figura 15. A simple vista puede llamar la atención que la tabla "Todo" tenga tantos campos, en vez de estar dividida en tablas más pequeñas que reduzcan la complejidad de la misma. El motivo de mantener la tabla "Todo" con todos esos campos es debido a que el cliente ha solicitado que se mantenga de esta forma, ya que desea que esa tabla almacene toda la información de carácter bibliográfico asociada a un registro. Aparte de la tabla principal "Todo", existen otras tres tablas, más pequeñas y sencillas de manejar, estas tablas recogen por un lado la relación de autores con sus publicaciones, las veces que un artículo ha sido citado, y la dirección asociada a un artículo.

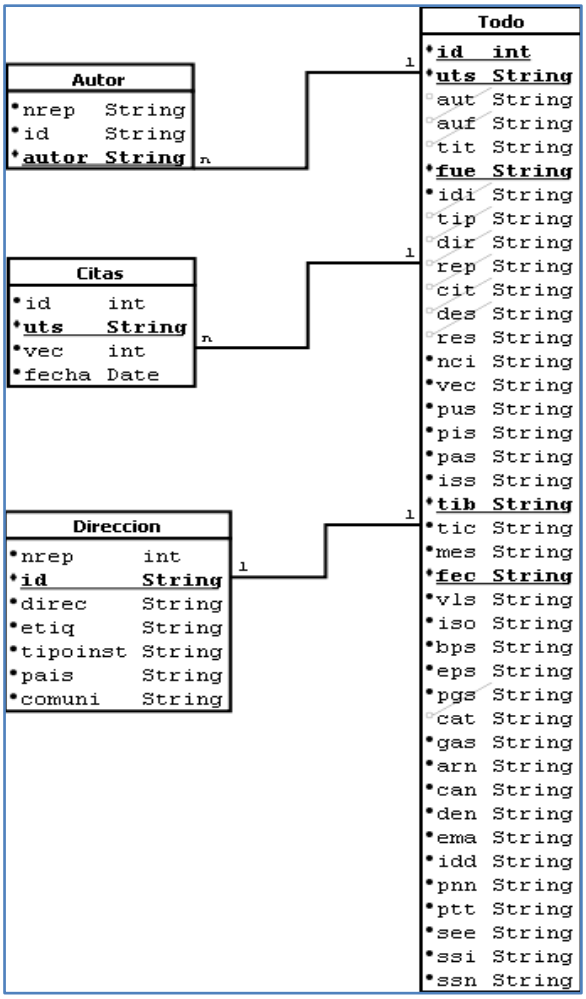

Figura 15: Diagrama E/R del modelo de datos

Modelo

Para diseñar el modelo se ha considerado dividir la capa en dos partes, una que contenga la lógica de negocio y otra que recoja el acceso a los datos. Para implementar la lógica de negocio se hace uso de ORM, técnica de programación para convertir datos entre los tipos utilizados en un lenguaje de programación orientado a objetos y los utilizados en una base de datos relacional. Para ello se implementarán una serie de POJOs, que representarán a todas las entidades significativas que forman parte del sistema. La aplicación contará con tantas entidades como tablas forman parte de la base de datos, por lo que cada entidad estará directamente relacionada con su tabla correspondiente. A continuación se describen los POJOs con que cuenta el sistema:

- **Autor:** representa al autor de una publicación, está identificado por su nombre completo y el uts (identificador impuesto por el WoK).
- **Citas:** hace referencia a las veces que un artículo ha sido citado, su valor varía a lo largo del tiempo, por lo que lleva asociada una fecha para poder estudiar las veces que el artículo ha sido citado a lo largo del tiempo.
- **Dirección:** almacena la información relacionada con la dirección asociada a un artículo, de forma que se identifica el país de la producción, el ámbito académico, etc.
- **Todo:** es la entidad más importante y compleja del sistema, recoge toda la información principal asociada a un artículo. Se podría haber considerado la posibilidad de dividirla en entidades más pequeñas para reducir su complejidad,

pero el cliente solicitó expresamente que se mantuviera de esta forma. Almacena entre otros datos, el autor del artículo, la revista donde se publicó, su ISSN (*International Standard Serial Number*, Número Internacional Normalizado de Publicaciones Seriadas).

Cada una de las entidades tiene asociado un DAO, para poder acceder a los datos. Los DAOs de los que hará uso el sistema se muestran a continuación en el diagrama de clases de la Figura 16.

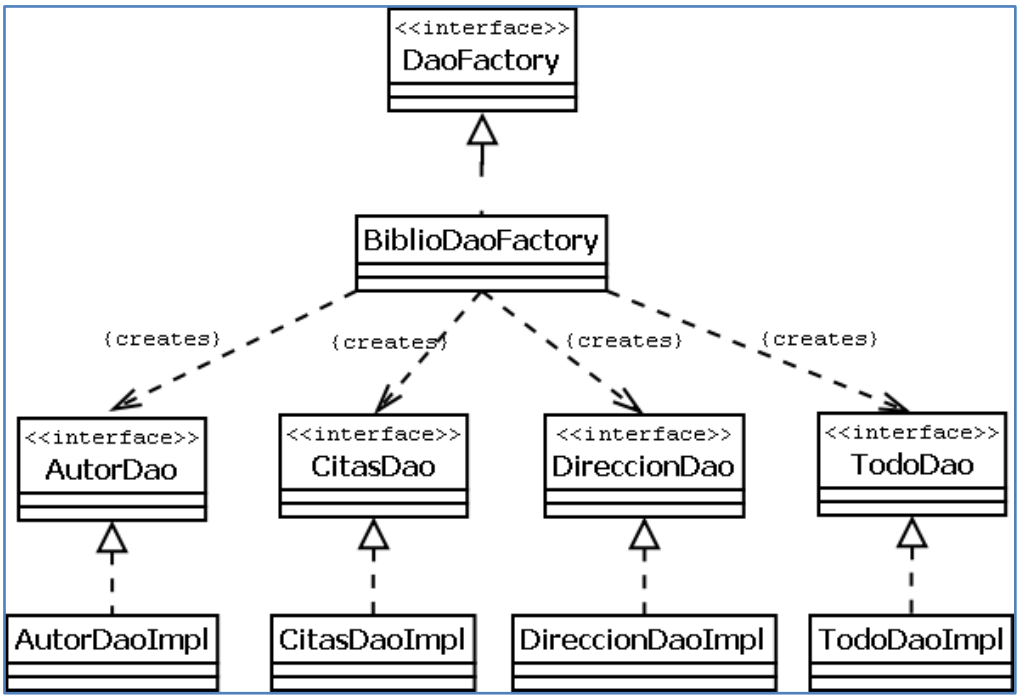

Figura 16: Diagrama de clases de los DAOs

Finalmente, para poder realizar distintas operaciones sobre los datos, hay que desarrollar una serie de servicios utilizando EJBs de sesión. Los servicios que debe ofrecer la aplicación son los siguientes:

- **Añadir autor (AddAutor):** añade un nuevo autor a la base de datos.
- **Añadir citas (AddCitas):** añade nuevas citas a la base de datos con la fecha del momento de la inserción.
- **Añadir dirección (AddDireccion):** inserta una nueva dirección asociada a un artículo a la base de datos.
- **Añadir a todo (AddTodo):** añade un nuevo registro a la tabla "todo".
- **Buscar autor (BuscarAutor):** devuelve a todos los autores que cumplan los criterios de búsqueda.
- **Buscar citas (BuscarCitas):** devuelve todas las citas que satisfagan los criterios de búsqueda especificados.
- **Buscar dirección (BuscarDireccion):** devuelve aquellas direcciones que cumplan con los criterios de búsqueda.
- **Buscar en todo (BuscarTodo):** devuelve los registros de la tabla "todo" que cumplan con las condiciones de búsqueda.
- **Modificar autor (ModificarAutor):** modifica los datos del autor con los nuevos datos indicados.
- **Modificar citas (ModificarCitas):** modifica la cita seleccionada con los nuevos datos de entrada.

- **Modificar dirección (ModificarDirección):** modifica los datos del registro de dirección con los nuevos datos indicados.
- **Modificar todo (ModificarTodo):** modifica los datos del registro de la tabla "todo" seleccionado.

Interfaz

La capa de la interfaz representa lo que el usuario va a ver y mediante la cual se va a comunicar e interactuar con el sistema. El objetivo de la interfaz es que sea intuitiva y sencilla de usar por lo que su diseño es de vital importancia. La Figura 17 muestra el diseño orientativo de lo que deberá ser la interfaz. Para llegar a este diseño de la interfaz, nos hemos basado en otras aplicaciones similares desarrolladas por el grupo DEI, ya que presentaban una gran sencillez y su uso era muy intuitivo. Además, se le ha presentado al usuario un prototipo orientativo donde se recogía la funcionalidad prevista para la aplicación y la distribución de los distintos elementos dentro de la interfaz, para contar con su opinión sobre el diseño de la interfaz. Estos prototipos iniciales se recogen en …

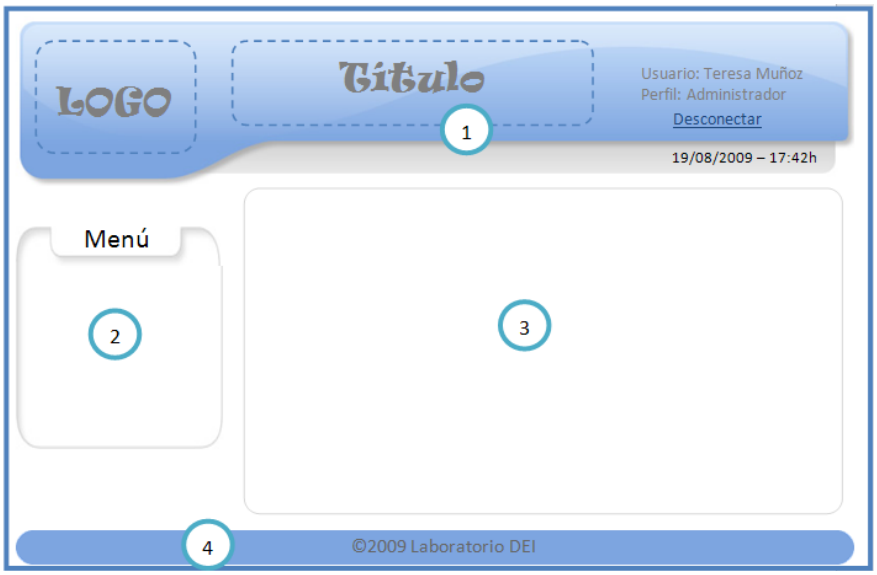

Figura 17: Diseño de la interfaz de usuario

En la Figura 17, puede verse que la interfaz se ha dividido en zonas atendiendo a la información que se va a mostrar en cada una de ellas. La zona 1 representa en encabezado de la página, donde aparecerán el logotipo de la aplicación, el nombre de la misma, el nombre y el perfil del usuario, la fecha y hora en el formato indicado y la posibilidad de desconectarse de la aplicación. La zona que aparece identificada con el número 2 representa el menú de la aplicación, donde estarán disponibles todas las operaciones que el usuario pueda realizar con la aplicación. La zona 3 es la zona principal de la aplicación, es donde aparecerán las distintas pantallas con formularios solicitando información al usuario o bien se mostrarán los datos que el usuario haya solicitado. Por último, la zona 4 representa el pie de la página, donde aparece el copyright y sirve de delimitador de la misma.

Uno de los principales problemas que se plantean en el desarrollo de la interfaz de usuario es establecer la navegabilidad del sistema por las distintas pantallas y opciones, de forma que resulte sencilla e intuitiva para el usuario. Para facilitar esta labor, se usan diagramas de navegación que ayudan a visualizar el comportamiento del usuario con el sistema, a través de las distintas páginas. El diagrama de navegación correspondiente a este sistema se puede ver en la Figura 18.

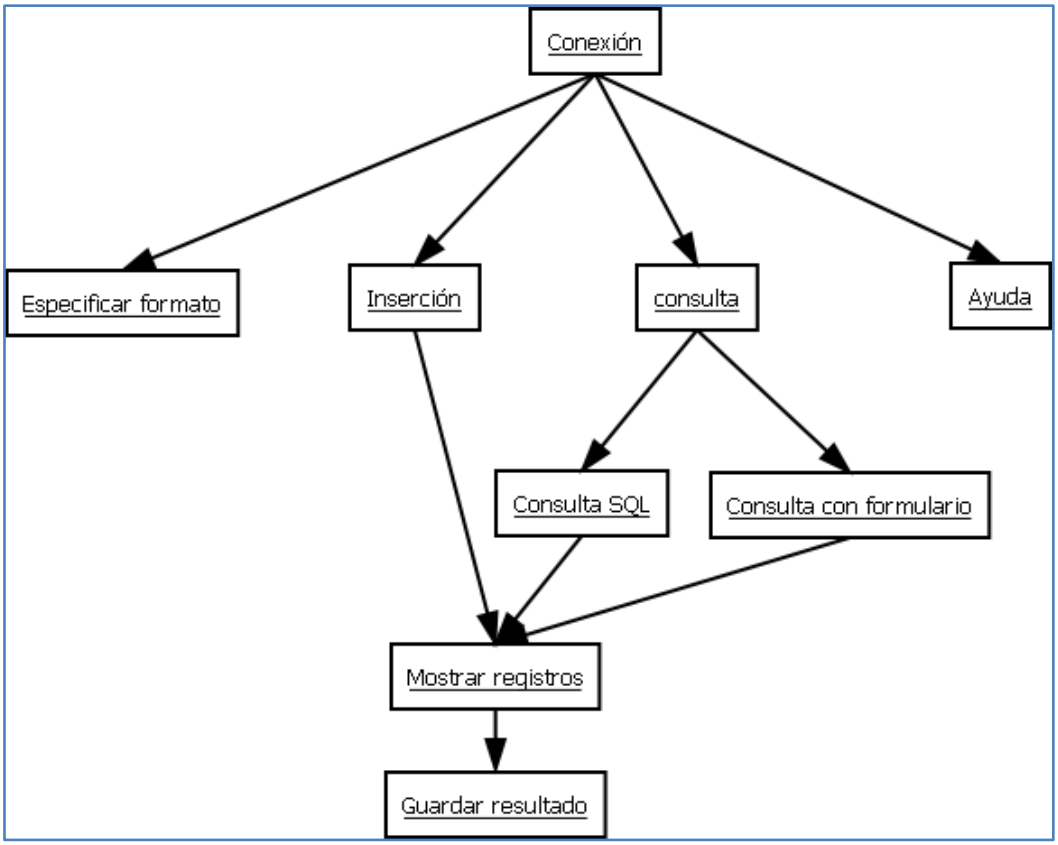

Figura 18: Diagrama de navegación

4.2.3 Implementación

Una vez finalizadas las fases de análisis y diseño del sistema, pasamos a la fase de implementación, donde se toman los requisitos y los productos de las fases anteriores y se desarrollan mediante las tecnologías elegidas. En esta sección vamos a centrar en las herramientas utilizadas para el desarrollo del proyecto, así como la organización del mismo.

Herramientas

En esta sección se describen las herramientas que se han utilizado para facilitar el desarrollo del proyecto. Estas herramientas se han escogido de forma que pudieran ser utilizadas en combinación con las tecnologías elegidas para la implementación de la aplicación.

- **Eclipse 3.4.2:** para el desarrollo del proyecto, se ha utilizado Eclipse [7], un IDE (Integrated Development Environment, Entorno de desarrollo integrado), para desarrollar aplicaciones en varios lenguajes, aunque en este caso, se ha optado por Java. Eclipse ofrece una gran funcionalidad que puede verse ampliada mediante plug-ins según las necesidades. El motivo de elegir Eclipse frente a otros IDEs de funcionalidad similar, ha sido que estoy más familiarizada con su uso, con lo cual no ha sido necesario invertir tiempo en el aprendizaje de la herramienta. Además, Eclipse se caracteriza por consumir pocos recursos durante la ejecución.
- **MySQL 5.0:** como SGBD se ha elegido MySQL, los motivos de esta elección son varios, pero el principal es el excelente rendimiento que ofrece frente a otros gestores de bases de datos gratuitos, lo que lo hace idóneo para su uso en combinación con una aplicación web. Por otro lado, el uso de MySQL está muy extendido, lo que significa que hay numerosa documentación disponible sobre su uso, lo cual es de gran ayuda en caso de que surja algún tipo de complicación.

- **JBoss 4.2:** se trata de un servidor de aplicaciones J2EE de código abierto e implementado en Java, lo que lo hace ser multiplataforma. El motivo de utilizar JBoss frente a otros servidores como pueda ser Apache Tomcat, es debido a que JBoss incluye la mayor parte de las librerías necesarias para el desarrollo de la aplicación, destacando especialmente su soporte de EJB.
- **Internet Explorer 8.0:** este navegador web producido por Microsoft, se utilizará durante la fase de pruebas de la aplicación, ya que es el navegador más usado actualmente por los usuarios.
- **Firefox 3.5.2:** durante la fase de pruebas también se empleará este navegador de código abierto, ya que es multiplataforma y también cuenta con numerosos usuarios actualmente.

Organización

Para el desarrollo de la aplicación, se ha dividido el código en tres proyectos, atendiendo a la naturaleza del mismo. De esta forma se ha separado el código de la parte web de la aplicación, del código correspondiente a EJB, de esta forma se facilita el desarrollo y el futuro mantenimiento de la aplicación. Los proyectos Eclipse que se han creado son los siguientes:

- **Biblio_WEB:** este proyecto agrupa todo el código relacionado con la parte web de la aplicación (la vista), incluyendo las páginas web que mostrará la vista de la aplicación, las hojas de estilo, las imágenes de las que hace uso la vista, las clases Java que implementan los componentes, etc.
- **Biblio_EJB:** este proyecto recoge todo el código correspondiente al resto de capas de la aplicación. Se encarga de recibir las peticiones del usuario, acceder a la base de datos y devolver a la vista (proyecto Biblio_WEB) los resultados pertinentes.
- **Biblio_EAR:** este proyecto tiene como finalidad agrupar los dos proyectos anteriores en uno para poder ejecutarlos simultáneamente y de manera coherente en el servidor. Recibe este nombre porque el archivo principal que permite el empaquetado de los otros módulos tiene formato EAR (Enterprise ARchive, archivo de empresa).

Una vez vistos y explicados los proyectos en que se ha dividido la aplicación, a continuación se va a describir cómo se han estructurado cada uno de ellos, dando cuenta de la funcionalidad que recoge cada uno de los paquetes en que se han dividido los proyectos.

El proyecto Biblio_WEB, recoge toda la funcionalidad en la carpeta *src/main*, que se divide en dos paquetes *java* y *webapp*. El paquete denominado *java*, recoge todas las clases .java necesarias para implementar la funcionalidad del sistema, se divide a su vez en tres paquetes principales: *es/uc3m/dei/biblio/auxiliar*, que recoge las clases java auxiliares para completar el funcionamiento de otras; *es/uc3m/dei/biblio/beans*, donde se almacenan todos los beans manejados de los que hace uso la aplicación; *es/uc3m/dei/biblio/components*, este paquete recoge todas las clases java necesarias para la implementación de los distintos componentes de interfaz de usuario que forman parte de la aplicación, se divide en subpaquetes para agrupar los componentes según su funcionalidad. El paquete *webapp* soporta la parte web propiamente dicha, dividiéndose en tres paquetes principales: *pages*, donde se almacenan las páginas .jsf que se mostrarán en el navegador del usuario; *resources*, aquí se recogen todos los recursos web necesarios para mostrar las páginas .jsf de forma correcta, incluyéndose las hojas de estilo css, las imágenes de las que hace uso la vista, etc.; *WEB-INF* recoge el fichero faces-config.xml donde se establecen las reglas de navegación y la lista de beans manejados, además se incluye la carpeta *lib*, donde se guardan las librerías de las que hace uso el proyecto web. La estructura completa del proyecto Biblio_WEB en paquetes puede verse en la Figura 19.

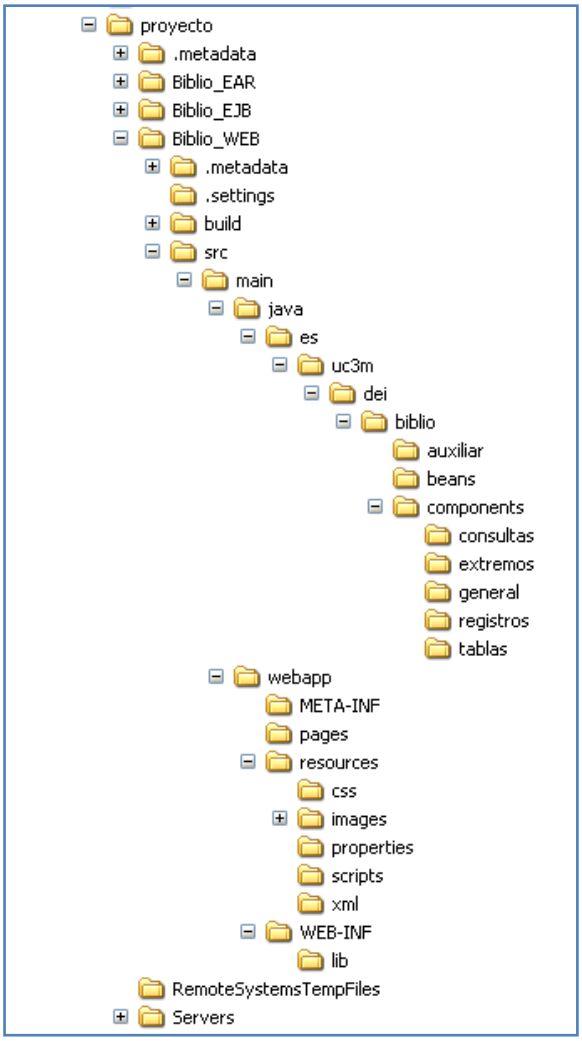

Figura 19: Estructura de paquetes proyecto Biblio_WEB

El proyecto Biblio_EJB recoge toda su funcionalidad en el paquete *ejbModule*. En el paquete *META-INF* se almacena el fichero persistence.xml donde se almacena la información relativa a la unidad de persistencia. Además, también contiene el fichero biblio-ds.xml, que recoge la información asociada a la base de datos que se va a utilizar y que deberá ser copiado en la carpeta del servidor durante la fase de despliegue del proyecto. Las clases que implementan la capa de persistencia de la aplicación se recogen en dos paquetes *es/uc3m/dei/biblio* y *es/uc3m/dei/fw2008*. El primero de ellos recoge los beans, daos y servicios propios de la aplicación a desarrollar, mientras que el segundo recoge las clases para el manejo de los mismos contenidas en el framework. La estructura completa de paquetes del proyecto Biblio_EJB puede verse en la Figura 20.

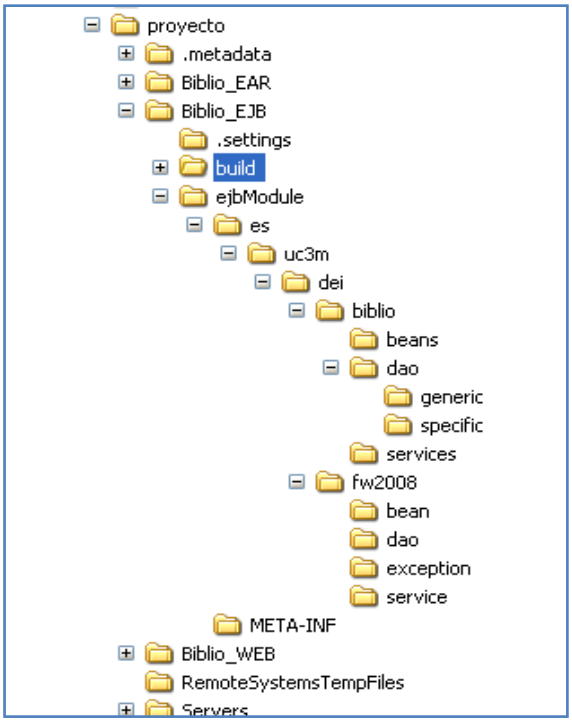

Figura 20: Estructura de paquetes proyecto Biblio_EJB

4.3 El producto del desarrollo

Tras finalizar el proceso de desarrollo, se obtiene como resultado una aplicación web que permite gestionar toda la información almacenada en la base de datos del grupo LEMI. Esta gestión abarca las operaciones principales de actualización de la base de datos y de consulta de la información almacenada en la misma. El sistema desarrollado se caracteriza por su sencillez en el manejo, además de lo intuitivo en el uso de la interfaz.

La ejecución de la aplicación es bastante sencilla, ya que a pesar de contar con tres proyectos diferentes (Biblio_WEB, Biblio_EJB y Biblio_EAR), es el proyecto Biblio_EAR el que engloba la funcionalidad de los otros dos, por lo que no es necesario ejecutar cada proyecto de manera independiente, sino que tan sólo hay que ejecutar el proyecto Biblio_EAR y la aplicación está desplegada.

Una vez desplegada la aplicación, el usuario ya puede empezar a utilizarla, para acceder a ella es suficiente con introducir la URL de donde esté alojada la aplicación en cualquier navegador web, aunque por el momento sólo se asegura su correcto funcionamiento en los navegadores Internet Explorer y Mozilla Firefox. Tras introducir la URL adecuada, se muestra al usuario una pantalla de inicio con un formulario para acceder a la aplicación, donde se solicitan su identificador de usuario y contraseña. Una vez introducidos correctamente estos datos, se abre la aplicación propiamente dicha, dando acceso al usuario a toda la funcionalidad para la que tenga permisos.

La aplicación permite al usuario actualizar la información almacenada en la base de datos, insertando nuevos registros a la misma o bien modificando los ya existentes. El usuario selecciona el fichero de entrada donde están todos los registros a incluir en la base de datos, así como el formato del fichero de entrada. Una vez seleccionado el fichero de entrada, la aplicación lee internamente y de forma completamente transparente al usuario cada uno de los registros contenidos en el fichero de origen. Por cada registro, la aplicación comprueba si existe previamente en la base de datos, en caso afirmativo, modifica el registro existente con los nuevos datos, en caso negativo lo inserta nuevo. En este proceso se rellenan todas las

tablas de la base de datos para dotarlas de la información correspondiente, además de guardarse una copia de qué registros han sido modificados y cuáles insertados para mostrar esta información posteriormente al usuario.

El usuario también puede realizar consultas sobre la base de datos de manera sencilla a través de la aplicación, permitiendo ésta establecer qué campos de la tabla elegida se quieren mostrar y rellenado el formulario donde basta con indicar qué condiciones debe cumplir la consulta, indicando por cada una de ellas si el texto introducido debe coincidir con el del campo, formar parte del mismo o no estar presente. El resultado de la consulta se muestra en forma de tabla.

Otra forma de realizar consultas sobre la base de datos a través de la aplicación es mediante sentencias SQL, de esta forma se pueden realizar consultas más complejas que impliquen la participación de varias tablas de la base de datos y no de una sólo. El resultado de la consulta efectuada se muestra en forma de tabla, pero si la sintaxis de la misma no es correcta, se muestra al usuario un mensaje con el tipo de error producido para que tenga información de cómo reescribir la consulta.

Para aumentar la versatilidad de la aplicación, ésta permite a los usuarios definir nuevos formatos de los ficheros de origen, para de esta forma recoger información bibliográfica no sólo de la Web of Knowledge, sino también de otras web que contengan información catalogada con los mismos campos pero distribuidos de manera diferente. El formato de origen se especifica indicando cómo están identificados cada uno de los campos de los registros, bien por etiquetas, saltos de página, etc. Además se debe especificar cómo se separan unos registros de otros.

Evaluación

5 Evaluación

La finalidad del capítulo de evaluación es demostrar la validez de la solución elaborada, verificando que la aplicación desarrollada resuelve los problemas expuestos en el planteamiento del problema y satisface los objetivos definidos en la introducción, cumpliendo por tanto con los requisitos y exigencias expuestos por el usuario. La evaluación se desarrolla a través de un conjunto de casos de prueba, que permiten verificar si la salida obtenida coincide con la esperada y así comprobar el correcto funcionamiento del sistema. En este capítulo se recoge el proceso de evaluación seguido para a continuación realizar un análisis de los resultados obtenidos.

5.1 Proceso de evaluación

El proceso de evaluación es el proceso mediante el cual se comprueba el correcto funcionamiento del sistema desarrollado, para ello se determinan y ejecutan una serie de pruebas para probar aspectos como la integridad de la base de datos, la funcionalidad del sistema y la interfaz de usuario.

5.1.1 Plan de pruebas

La aplicación se ha evaluado en función de un conjunto de casos de prueba mediante los cuales se determinará si la aplicación cumple con los requisitos especificados por el cliente. Por ello, se determinará al menos un caso de prueba por cada requisito identificado en la fase de análisis. Estos casos de prueba se caracterizan por contar con una entrada conocida y una salida esperada, que son definidas antes de que se ejecute la prueba, tras realizar la prueba se debe comprobar que la salida obtenida coincide con la esperada y por lo tanto el sistema cumple con la funcionalidad establecida. A continuación se describe la funcionalidad que deberán comprobar los distintos casos de prueba:

- La base de datos almacena los siguientes datos de los usuarios registrados: nombre y apellidos, perfil, identificador y contraseña.
- La aplicación sólo puede ser accedida por usuarios registrados.
- Sólo los administradores pueden añadir nuevos usuarios al sistema.
- Sólo los administradores pueden eliminar usuarios previamente registrados en el sistema.
- Sólo los administradores pueden modificar los datos de los usuarios registrados en el sistema.
- Tras pulsar el botón de desconexión, el usuario previamente conectado sale de la aplicación.
- El sistema permite ejecutar consultas mediante formularios.
- El sistema permite efectuar consultas mediante sentencias SQL.
- El sistema muestra de manera correcta el resultado de la consulta ejecutada en forma de tabla.
- La aplicación permite al usuario especificar el formato de entrada de los archivos de inserciones.
- La aplicación ejecuta las inserciones y modificaciones de manera correcta.
- El sistema muestra información sobre el proceso de inserción y/o modificación.
- La aplicación permite recuperar en forma de tabla los registros insertados y/o modificados.
- La aplicación permite guardar en fichero los registros mostrados en forma de tabla.

- La aplicación ofrece un sistema de ayuda.

5.1.2 Casos de prueba

En esta sección se describen los distintos casos de prueba ejecutados para comprobar el correcto funcionamiento de la aplicación. Para cada caso de prueba identificado y ejecutado, se recoge en forma de tabla la siguiente información:

- **Identificador:** identifica de manera unívoca al caso de prueba.
- **Propósito:** breve descripción del propósito de la prueba así como la funcionalidad que se comprueba.
- **Requisitos vinculados:** identifica los requisitos funcionales que dan lugar al caso de prueba.
- **Dependencias:** indica la dependencia existente con otros casos de prueba que deben ser ejecutados previamente.
- **Especificaciones de entrada:** descripción de los datos de entrada necesarios para ejecutar la prueba.
- **Especificaciones de salida:** descripción de la salida esperada.
- **Resultado:** tomará los valores correcto o fallido, dependiendo de si el resultado de la prueba coincide con la salida esperada o no.

A continuación se describen los casos de prueba que se han ejecutado para comprobar el funcionamiento de la aplicación, así como el resultado obtenido en cada uno de ellos.

CP-01	
Propósito	Comprobar que la base de datos almacena los siguientes datos de los usuarios registrados: nombre y apellidos, perfil, identificador y contraseña.
Requisitos vinculados	RF-01
Dependencias	-
Especificaciones de entrada	Acceder a la base de datos del sistema y consultar la tabla donde se recogen los datos de los usuarios.
Especificaciones de salida	La tabla de usuarios recoge para cada uno de los usuario almacenados su nombre y apellidos, perfil, identificador y contraseña.
Resultado	Correcto

Tabla 76: Descripción caso de prueba CP-01

CP-02	
Propósito	Comprobar que la aplicación puede ser accedida por usuarios previamente registrados.
Requisitos vinculados	RF-002
Dependencias	CP-01
Especificaciones de entrada	Acceder a la página de acceso al sistema e ingresar el identificador y la contraseña de un usuario registrado.
Especificaciones de salida	Se muestra la página de inicio para usuarios conectados al sistema, apareciendo el nombre y apellidos, así como el perfil del usuario que se ha conectado.
Resultado	Correcto

Tabla 77: Descripción caso de prueba CP-02

CP-03	
Propósito	Comprobar que la aplicación no puede ser accedida por usuarios que no hayan sido previamente registrados.
Requisitos vinculados	RF-002
Dependencias	CP-01
Especificaciones de entrada	Acceder a la página de acceso al sistema e ingresar el identificador y la contraseña que no coincidan con los de un usuario registrado.
Especificaciones de salida	Se muestra de nuevo la pantalla de acceso al sistema, esta vez junto con el mensaje de "El usuario y/o la contraseña introducidos no son correctos".
Resultado	Correcto

Tabla 78: Descripción caso de prueba CP-03

CP-04	
Propósito	Comprobar que los usuarios con perfil de administrador pueden añadir nuevos usuarios al sistema.
Requisitos vinculados	RF-003
Dependencias	CP-02
Especificaciones de entrada	Escoger la opción "Añadir nuevo usuario", introducir los datos del nuevo usuario requerido, salir del sistema y volver a conectarse con el identificador y la contraseña del nuevo usuario.
Especificaciones de salida	Se muestra la página de inicio para usuarios conectados al sistema, apareciendo el nombre y apellidos, así como el perfil del nuevo usuario.
Resultado	Correcto

Tabla 79: Descripción caso de prueba CP-04

CP-05	
Propósito	Comprobar que los usuarios con perfil de administrador pueden modificar los datos de usuarios registrados en el sistema.
Requisitos vinculados	RF-004
Dependencias	CP-02
Especificaciones de entrada	Acceder al sistema como usuario administrador, escoger la opción "Modificar usuario", modificar los datos del usuario requerido, salir del sistema y volver a conectarse con el identificador y la contraseña del usuario modificado.
Especificaciones de salida	Se muestra la página de inicio para usuarios conectados al sistema, apareciendo el nombre y apellidos, así como el perfil adaptados a los cambios establecidos por el administrador.
Resultado	Correcto

Tabla 80: Descripción caso de prueba CP-05

CP-06	
Propósito	Comprobar que los usuarios con perfil de administrador pueden eliminar usuarios registrados en el sistema.
Requisitos vinculados	RF-005
Dependencias	CP-02, CP-03
Especificaciones de	Acceder al sistema como usuario administrador, escoger la opción

entrada	"Eliminar usuario", seleccionar el usuario requerido, salir del sistema y volver a conectarse con el identificador y la contraseña del usuario modificado.
Especificaciones de salida	Se muestra de nuevo la pantalla de acceso al sistema, esta vez junto con el mensaje de "El usuario y/o la contraseña introducidos no son correctos".
Resultado	Correcto

Tabla 81: Descripción caso de prueba CP-06

CP-07	
Propósito	Comprobar que los usuarios que no tienen perfil de administrador no pueden acceder a las opciones de añadir, modificar y eliminar usuarios.
Requisitos vinculados	RF-003, RF-004, RF-005
Dependencias	CP-02
Especificaciones de entrada	Acceder al sistema como usuario no administrador.
Especificaciones de salida	En el menú de opciones que aparece a la izquierda de la pantalla no aparecen las opciones asociadas a la gestión de usuarios.
Resultado	Correcto

Tabla 82: Descripción caso de prueba CP-07

CP-08	
Propósito	Comprobar que el usuario sale de manera correcta del sistema tras pulsar el botón de desconexión.
Requisitos vinculados	RF-006
Dependencias	CP-02
Especificaciones de entrada	Una vez conectado al sistema el usuario, pulsar el botón "Desconexión".
Especificaciones de salida	Se muestra la pantalla de acceso al sistema.
Resultado	Correcto

Tabla 83: Descripción caso de prueba CP-08

CP-09	
Propósito	Comprobar que el sistema permite efectuar consultas sencillas sobre tablas a través de formularios.
Requisitos vinculados	RF-007
Dependencias	CP-02
Especificaciones de entrada	Una vez conectado al sistema el usuario, seleccionar la opción "Consultas con formularios" y seguir los pasos indicados por el sistema.
Especificaciones de salida	La aplicación muestra correctamente los formularios necesarios para efectuar la consulta.
Resultado	Correcto

Tabla 84: Descripción caso de prueba CP-09

CP-010	
Propósito	Comprobar que el sistema permite efectuar consultas mediante sentencias SQL.

Requisitos vinculados	RF-008
Dependencias	CP-02
Especificaciones de entrada	Una vez conectado al sistema el usuario, seleccionar la opción "Consultas SQL" e incluir la sintaxis de la consulta a ejecutar.
Especificaciones de salida	La aplicación muestra correctamente los formularios necesarios para efectuar la consulta.
Resultado	Correcto

Tabla 85: Descripción caso de prueba CP-010

CP-011	
Propósito	El sistema muestra de manera correcta el resultado de la consulta ejecutada en forma de tabla.
Requisitos vinculados	RF-009
Dependencias	CP-09, CP-10
Especificaciones de entrada	Ejecutar una de las dos opciones de consulta, de forma que se obtengan como resultado varios registros.
Especificaciones de salida	Se muestra una tabla que recoge los distintos registros que compone el resultado de la consulta.
Resultado	Correcto

Tabla 86: Descripción caso de prueba CP-011

CP-012	
Propósito	Comprobar que la aplicación permite al usuario especificar el formato de entrada de los archivos de inserciones.
Requisitos vinculados	RF-010
Dependencias	CP-02
Especificaciones de entrada	Una vez conectado el usuario al sistema, escoger la opción "Especificar formato", rellenar los campos de los formularios y escoger la opción "Insertar registros".
Especificaciones de salida	El nuevo formato de entrada aparece en la lista de formatos de archivos.
Resultado	Correcto

Tabla 87: Descripción caso de prueba CP-012

CP-013	
Propósito	Comprobar que la aplicación permite efectuar el proceso de inserción y modificación de registros.
Requisitos vinculados	RF-011
Dependencias	CP-02, CP-012
Especificaciones de entrada	Una vez conectado el usuario al sistema, escoger la opción "Insertar registros".
Especificaciones de salida	Se muestra la pantalla donde el usuario puede escoger el formato de entrada del fichero entre los que ha definido, así como especificar el fichero de entrada que contiene los registros.
Resultado	Correcto

Tabla 88: Descripción caso de prueba CP-013

CP-014	
Propósito	Comprobar que el sistema ofrece información sobre el resultado del proceso de inserción y actualización de registros.

Requisitos vinculados	RF-012
Dependencias	CP-013
Especificaciones de entrada	Una vez conectado el usuario al sistema, escoger la opción “Insertar registros”, seleccionar el formato del fichero, así como el fichero de entrada y pulsar “Aceptar”.
Especificaciones de salida	Se muestra una pantalla con una serie de enlaces indicando el número de archivos insertados, modificados y de errores.
Resultado	Correcto

Tabla 89: Descripción caso de prueba CP-014

CP-015	
Propósito	Comprobar que la aplicación permite recuperar en forma de tabla los registros insertados.
Requisitos vinculados	RF-013
Dependencias	CP-014
Especificaciones de entrada	Una vez conectado el usuario al sistema, escoger la opción “Insertar registros”, seleccionar el formato del fichero, así como el fichero de entrada y pulsar “Aceptar”. Pulsar sobre el enlace que indica el número de registros insertados.
Especificaciones de salida	Se muestra una pantalla con una tabla que recoge los registros que han sido insertados nuevos al sistema.
Resultado	Correcto

Tabla 90: Descripción caso de prueba CP-015

CP-016	
Propósito	Comprobar que la aplicación permite recuperar en forma de tabla los registros modificados.
Requisitos vinculados	RF-013
Dependencias	CP-014
Especificaciones de entrada	Una vez conectado el usuario al sistema, escoger la opción “Insertar registros”, seleccionar el formato del fichero, así como el fichero de entrada y pulsar “Aceptar”. Pulsar sobre el enlace que indica el número de registros modificados.
Especificaciones de salida	Se muestra una pantalla con una tabla que recoge los registros que han sido modificados en el sistema.
Resultado	Correcto

Tabla 91: Descripción caso de prueba CP-016

CP-017	
Propósito	Comprobar que la aplicación guardar en forma de fichero los registros mostrados en forma de tabla.
Requisitos vinculados	RF-014
Dependencias	CP-11, CP-015, CP-016
Especificaciones de entrada	Una vez el usuario se encuentra en la pantalla donde se muestran los registros en forma de tabla tras efectuar una inserción o consulta, escoge la opción “Guardar en fichero”.
Especificaciones de salida	Los registros de la tabla se han guardado en un fichero en el sistema de archivos del usuario.
Resultado	Correcto

Tabla 92: Descripción caso de prueba CP-017

CP-018	
Propósito	Comprobar que la aplicación ofrece un sistema de ayuda al usuario.
Requisitos vinculados	RF-015
Dependencias	CP-02
Especificaciones de entrada	Tras haberse conectado, el usuario pulsa sobre la opción “Ayuda”.
Especificaciones de salida	La aplicación muestra una página donde se recoge información sobre como ejecutar la funcionalidad que ofrece el sistema.
Resultado	Correcto

Tabla 93: Descripción caso de prueba CP-018

5.2 Análisis de resultados

Tras la ejecución de los distintos casos de prueba y la superación satisfactoria de todos ellos, se ha comprobado que la aplicación cumple con la funcionalidad requerida y exigida por el usuario, de forma que se satisfacen todos los requisitos funcionales establecidos durante la fase de análisis de este proyecto. Por lo tanto, se puede asegurar que el sistema cumple con la siguiente funcionalidad:

- El acceso al sistema está restringido a aquellos usuarios previamente registrados.
- El sistema almacena los datos de los usuarios registrados necesarios para su conexión al sistema.
- La aplicación ofrece un sistema de gestión de usuarios que sólo puede ser gestionada por los usuarios con perfil de administrador.
- El sistema permite insertar nuevos registros a la base de datos, así como modificar los ya existentes, ofreciendo información sobre el proceso.
- La aplicación ofrece dos sistemas de consulta, mediante formularios y mediante secuencias SQL.
- El sistema permite especificar el formato de entrada de los ficheros con los registros a insertar y/o modificar.
- El resultado de las consultas e inserciones puede ser almacenado en forma de archivo.
- La aplicación ofrece un sistema de ayuda para facilitar su manejo.

Conclusión

6 Conclusión

En este capítulo se recogen las conclusiones extraídas de la realización de este Proyecto Fin de Carrera. Dentro de estas conclusiones se hace balance de las aportaciones que ofrece este proyecto, se perfilan algunos de los trabajos futuros pueden desarrollarse a partir del mismo, los problemas a los que se ha tenido que hacer frente durante su desarrollo y se concluye con mi visión personal acerca de todo el proceso.

6.1 Aportaciones realizadas

Este proyecto se plantea como solución a los múltiples problemas que presentaba el proceso de gestión y actualización de la información bibliográfica almacenada en la base de datos del LEMI. A continuación se describen las aportaciones que supone este proyecto:

- **Simplificación del proceso de actualización de la información:** el proceso de inserción y actualización de la información bibliográfica almacenada en la base de datos se ha visto simplificado gracias a la aplicación desarrollada. Este proceso ya no depende de complicadas sentencias Perl, sino que basta con que el usuario seleccione el fichero con la información a actualizar y el formato del mismo, y la aplicación internamente se encarga de llevar a cabo todo el proceso de manera transparente al usuario.
- **Información del proceso de actualización:** hasta ahora, el proceso de actualización era algo opaco al usuario, que no sabía si se había realizado con éxito, ni qué cantidad de registros se añadían nuevos a la base de datos, ni cuáles eran modificados con respecto a la información previamente almacenada. Ahora, tras efectuar las inserciones correspondientes se muestra información sobre cómo ha ido el proceso, indicando el número de registros actualizados, insertados y que han producido algún tipo de incidencia. Además, la aplicación permite ver cuáles han sido exactamente los archivos modificados o insertados y guardarlos en un fichero para su posterior recuperación.
- **Sencillez en las consultas a la base de datos:** la aplicación permite realizar consultas sobre la base de datos de manera sencilla e intuitiva, mediante formularios. Además, se puede seleccionar qué campos de la tabla se mostrarán y guardar en archivo el resultado de las distintas consultas para facilitar su recuperación posteriormente. Por otro lado, la aplicación también facilita el proceso de ejecutar consultas más complejas, mediante el uso de sentencias SQL.
- **Independencia de formato de entrada:** el sistema desarrollado permite insertar y modificar los registros de la base de datos con independencia del origen de los mismos y por tanto del formato del archivo que los contiene. Esto es debido a que la aplicación ofrece un sistema para especificar el formato en que se distribuyen los registros en el fichero de entrada, de tal forma que una vez se especifica un formato no es necesario volverlo a especificar en futuras inserciones.
- **Aplicación sencilla e intuitiva:** uno de los mayores inconvenientes que presentaba el antiguo sistema, era la dificultad de su uso, lo que lo restringía a usuarios con conocimientos previos en su manejo. Esta aplicación ha sido diseñada de tal forma que su utilización resultara intuitiva y sencilla para todo tipo de usuarios con conocimientos básicos de informática. Para asegurar la satisfacción del cliente con el diseño de la aplicación, se presentaron una serie de prototipos de la interfaz que fueron aprobados por el cliente.

6.2 Trabajos futuros

La aplicación desarrollada no es una aplicación cerrada, sino que está abierta a futuras ampliaciones para adaptarse a las nuevas necesidades que puedan surgir dentro del grupo LEMI o bien para contemplar aquellas, que por falta de tiempo y para no desarrollar un proyecto de tamaño desorbitado se han dejado fuera. Debido a las tecnologías empleadas y al diseño del proyecto, cualquier futura ampliación que suponga la inclusión de nuevos elementos no supondrá un gran esfuerzo y podrá ser fácilmente integrado con el resto de la aplicación hasta ahora desarrollada.

Una de las principales necesidades que se plantean desde el LEMI es la automatización no sólo del proceso de inserción de registros a su base de datos, sino también el proceso de descarga de esos registros desde la base de datos del WoK, ya que hasta ahora se efectúa a mano, pudiendo descargar un máximo de 500 registros en cada ocasión. El objetivo sería automatizar este proceso de tal forma que no tuviera que intervenir el factor humano para la descarga total de todos los archivos e integrarlo con el proceso de inserción que ahora mismo ofrece la aplicación desarrollada.

Otra posible ampliación, sería la integración de las distintas operaciones que actualmente se realizan sobre la base datos con fines métricos y de investigación dentro de la aplicación. Así, las operaciones más frecuentes podrían estar preestablecidas en el menú de la aplicación para facilitar su ejecución.

También podría incluirse a la aplicación una herramienta desarrollada para calcular el cuartil asociado a una publicación, necesario para calcular el complemento de retribución a los profesores de la universidad y que trabaja con los datos almacenados en la misma base de datos que la aplicación. Esta herramienta ya está desarrollada, por lo que simplemente habría que integrarla con la aplicación web, de forma que su uso fuera más sencillo, además de agrupar operaciones sobre la base de datos en una misma aplicación.

Se podría agregar también a la funcionalidad del sistema, la elaboración de distintas estadísticas y gráficas, para evaluar la producción científica en un intervalo de tiempo determinado, por universidades o por autor, complementando de esta forma la labor desarrollada por el grupo de estudios métricos de manera visual y sencilla.

Por otro lado, la herramienta desarrollada es en definitiva un sistema gestor de bases de datos por lo que podría aplicarse con ciertas modificaciones a la gestión de otras bases de datos que actualmente supongan un proceso complicado de actualización y que mediante esta herramienta se vería simplificado.

6.3 Problemas encontrados

El desarrollo de este proyecto no ha estado exento de problemas, los cuales se han centrado básicamente en el desconocimiento inicial de las tecnologías que se han utilizado y en la escasez de tiempo disponible para cumplir con la fecha prevista de entrega.

Antes de empezar con este proyecto tenía un total desconocimiento sobre qué eran las tecnologías JSF y EJB y para qué servían. Además, no había tenido la oportunidad de trabajar con el framework del grupo con anterioridad al desarrollo de este proyecto, por lo que tampoco conocía qué me podía aportar en el desarrollo del mismo ni cómo se manejaba. Por este motivo, he tenido que emplear mucho tiempo, más del inicialmente planeado, en el estudio y aprendizaje tanto de las tecnologías mencionadas como del framework, hasta que por fin he conseguido integrarlos y que trabajaran de forma conjunta en el proyecto. Uno de

los principales problemas que he tenido con el framework del grupo ha sido la escasa documentación existente sobre el mismo, lo que en un principio me impedía trabajar con autonomía y tener que depender de la ayuda y explicaciones de los miembros del grupo que habían contribuido a su desarrollo. Por otro lado, aunque el framework cuenta con una gran cantidad de componentes desarrollados que facilitan el trabajo en gran medida ya que sólo hay que integrarlos con el resto del proyecto, también ha sido necesario diseñar y añadir otros nuevos, como es el caso de las tablas, con los que hasta ese momento no contaba el framework y que eran indispensables para el desarrollo del proyecto. Otro de los problemas tecnológicos al que he tenido que hacer frente ha sido el entender cómo se relacionaban los beans manejados, con los servicios, DAOs, POJOs, etc., ya que en un principio me resultaba caótico e inexplicable que hubiera que dar tantos pasos para poder acceder a la base de datos, hasta que por fin entendí las ventajas que suponía el uso de estas estructuras y la versatilidad que aportaban, sobre todo a la hora de hacer cambios y adaptaciones.

El otro gran problema al que he tenido que hacer frente ha sido el tiempo. Desde un principio se ha planteado una planificación muy optimista, contando con tan sólo tres meses para la realización total del proyecto, incluyendo tanto el desarrollo de la aplicación como la redacción de la memoria. Por este motivo he tenido que dedicar un gran esfuerzo para tratar de cumplir con los plazos previstos y que esto no afectara a la calidad ni de la aplicación ni de la memoria.

Otra de las dificultades surgidas, pero que no supuso tanto esfuerzo como las anteriores, fue conseguir disponer todos los componentes en las páginas de la forma deseada y acordada con el cliente. Esto se hizo mediante hojas de estilo css, pero al principio dieron bastantes problemas, ya que existían estilos predefinidos en algunos componentes del framework y entraban en conflicto con los que se definían con las hojas css. Finalmente, y con mucha paciencia, se consiguió distribuir todos los elementos dentro de la página tal y como se había diseñado (4.2.2).

6.4 Opiniones personales

El desarrollo de este proyecto me ha supuesto un gran esfuerzo y dedicación para conseguir acabarlo a tiempo y aún así mantener la calidad que un proyecto de estas características precisa. Su realización me ha aportado conocimientos no sólo en el manejo de nuevas tecnologías, sino también en el desarrollo de proyectos (especialmente en la gestión del tiempo) y sobre todo en el trabajo a nivel individual.

Hasta ahora, durante la carrera, la mayor parte de las prácticas que he tenido que realizar han sido siempre en parejas, y si la práctica suponía un gran esfuerzo, se realizaban en grupo. El Proyecto Fin de Carrera, sin embargo, se desarrolla individualmente, sin depender de otros compañeros que te puedan ayudar y a los que rendir cuentas, lo que me ha permitido aprender a superar los problemas, tomar decisiones y planificar el tiempo por mí misma.

En cuanto al tema del tiempo, como ya se ha explicado en la sección anterior, ha supuesto un gran reto el realizar el proyecto en un periodo de tiempo tan ajustado. Sin embargo, gracias a ello, he aprendido a organizar mejor el tiempo disponible, planificar con anterioridad las tareas a desarrollar y cumplir los plazos establecidos con independencia de las horas diarias que tuviera que dedicarle al proyecto.

El uso de tecnologías bastante novedosas para el desarrollo de aplicaciones web, me ha supuesto algunos problemas, pero a posteriori ha resultado ventajoso, ya que me ha permitido adquirir conocimientos sobre tecnologías web que tienen mucho futuro por delante y que presentan numerosas ventajas en cuanto a versatilidad y reutilización frente a las tecnologías

más tradicionales. Lo mismo ha ocurrido con el uso del framework, que aunque en un principio ha requerido bastante tiempo el aprender a manejarlo, finalmente ha merecido la pena su uso, ya que si hubiera tenido que desarrollar todos los componentes que he utilizado en el proyecto yo misma, habría sido prácticamente imposible mantener la fecha de entrega prevista.

La elaboración de la memoria me ha permitido ampliar mis conocimientos sobre la documentación de proyectos y la importancia de que ésta esté explicada con claridad. También he aprendido a investigar y hacer una comparación entre distintos sistemas y tecnologías y a documentarlo con la bibliografía adecuada.

En resumen, la elaboración de este proyecto ha supuesto un gran esfuerzo y dedicación, pero también ha supuesto la adquisición de numerosos conocimientos muy útiles para el desarrollo de futuros proyectos y trabajos de investigación.

Bibliografía

7 Bibliografía

Referencias utilizadas para la realización del trabajo:

[1]. Adrian Holovaty, Jacob Kaplan-Moss. The Definitive Guide to Django: Web Development Done Right. Apress – 2007
[2]. Aplicación Web, recuperado el 17/07/09, http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web
[3]. Arquitectura del Software, recuperado el 17/07/09, http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html
[4]. Caso de uso, recuperado el 20/08/09, http://www.mastermagazine.info/termino/4184.php
[5]. Ciclo de vida del software, recuperado el 20/08/09, http://es.kioskea.net/contents/genie-logiciel/cycle-de-vie.php3
[6]. Definición de Framework, recuperado el 20/07/09, http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/capitulo2.pdf
[7]. Eclipse, recuperado el 21/08/09, http://www.eclipse.org/home/newcomers.php
[8]. Enterprise Java Beans (EJB), recuperado el 03/08/09, http://java.sun.com/products/ejb
[9]. Gestión de Proyectos, recuperado el 06/08/09, http://www.knoow.net/es/cieeconcom/gestion/gestiondeproyectos.htm
[10]. James Turner, Kevin Bedell. Struts Kick Start. SamsPub – 2002
[11]. Java Data Objects (JDO), recuperado el 03/08/09, http://java.sun.com/jdo/
[12]. Java Database Connectivity (JDBC), recuperado el 06/08/09, http://java.sun.com/products/jdbc/overview.html
[13]. Java Enterprise Edition (Java EE), recuperado el 16/08/09, http://java.sun.com/javaee/
[14]. Ley Orgánica de Protección de Datos (LOPD), recuperado el 23/08/09, http://www.boe.es/aeboe/consultas/bases_datos/doc.php?coleccion=iberlex&id=2008/00979
[15]. MySQL, recuperado el 27/07/09, http://www.mysql.com/
[16]. OLE DB, recuperado el 06/08/09, http://msdn.microsoft.com/en-us/library/ms722784%28VS.85%29.aspx
[17]. Open Database Connectivity, recuperado el 06/08/09, http://msdn.microsoft.com/en-us/library/ms710252%28VS.85%29.aspx
[18]. Oracle, recuperado el 27/07/09, http://www.oracle.com/global/es/products/database/index.html
[19]. Oracle Data Provider for .NET (ODB .NET), recuperado el 06/08/09, http://www.oracle.com/technology/tech/windows/odpnet/index.html
[20]. Patrón Modelo-Vista-Controlador, recuperado el 17/07/09, http://es.wikipedia.org/wiki/Modelo_Vista_Controlador
[21]. Perl, recuperado el 25/08/09, http://www.perl.org/about.html
[22]. Project Management Institute (PMI). A guide to the project management body of knowledge. USA – 2000
[23]. PostgreSQL, recuperado el 27/07/09, http://www.postgresql.org/about/

[24]. Restricciones de MySQL, recuperado el 27/07/09, http://dev.mysql.com/doc/refman/5.0/es/restrictions.html

[25]. RichFaces, recuperado 16/08/09, http://www.jboss.org/richfaces

[26]. Sistemas Gestores de Bases de Datos, recuperado el 27/07/09, http://en.wikipedia.org/wiki/Database_management_system

[27]. SQLJ, recuperado el 06/08/09, http://www.iso.org/iso/catalogue_detail.htm?csnumber=34137

[28]. SQL/XML, recuperado el 06/08/09, http://www.oracle.com/technology/tech/xml/xquery/sqlxml/index.html

[29]. S. Connell. Desarrollo y Gestión de Proyectos Informáticos. McGraw-Hill Iberoamericana – 1997

[30]. Structured Query Language (SQL), recuperado el 06/08/09, http://www.w3schools.com/sql/sql_intro.asp

[31]. Ueli Wahli, Gabriel Cohen et al. WebSphere Studio 5.1.2 JavaServer Faces and Service Data Objects. IBM RedBooks – 2004

[32]. Web of Knowledge, recuperado el 29/07/09, http://www.accesowok.fecyt.es/info/productos.html

[33]. WebDAV, recuperado el 06/08/09, http://www.webdav.org/

[34]. XQuery, recuperado el 06/08/09, http://www.w3.org/TR/xquery

Anexo I: Control de versiones

Anexo I. Control de versiones

En esta sección se incluyen las diferentes versiones que se han realizado de la memoria, para poder hacer un seguimiento de las distintas modificaciones a las que se ha sometido la memoria y entender mejor qué aspectos ha sido necesario mejorar y dedicarles mayor esfuerzo.

Página de estado			
Título del documento	Desarrollo de una aplicación web para la gestión de información bibliográfica		
Autor	Mª Teresa Muñoz-Reja Herrero		
Tutor	Ignacio Aedo Cuevas		
Historia del documento			
Versión	**Fecha**	**Apartado**	**Razón de cambio**
1.0	06/09/2009	Todos	Versión preliminar

Tabla 94: Control de versiones

Anexo II: Seguimiento de Proyecto

Anexo II. Seguimiento de proyecto fin de carrera

Debido a la necesidad de tener que presentar este proyecto en septiembre de 2009 y no poder demorar esta fecha bajo ningún concepto, ha resultado indispensable fijar una planificación adecuada, que se adaptara a las necesidades del proyecto y tratar de seguirla con la mayor fidelidad posible. En esta sección se presentan tanto la planificación inicial como la real, y se hace una comparativa de ambas.

Planificación inicial

A continuación se muestra la planificación que se estimó al comienzo del proyecto, contando con que la fecha de entrega aproximada del proyecto sería a mediados de septiembre y que esta fecha como mucho podría posponerse una o dos semanas. Se puede ver que para la fase de desarrollo de la aplicación se estimó inicialmente una dedicación de mes y medio, sin embargo, esta dedicación es parcial ya que durante el mismo tiempo se estimó que se podría ir avanzando la memoria. La planificación inicial se recoge Figura 21.

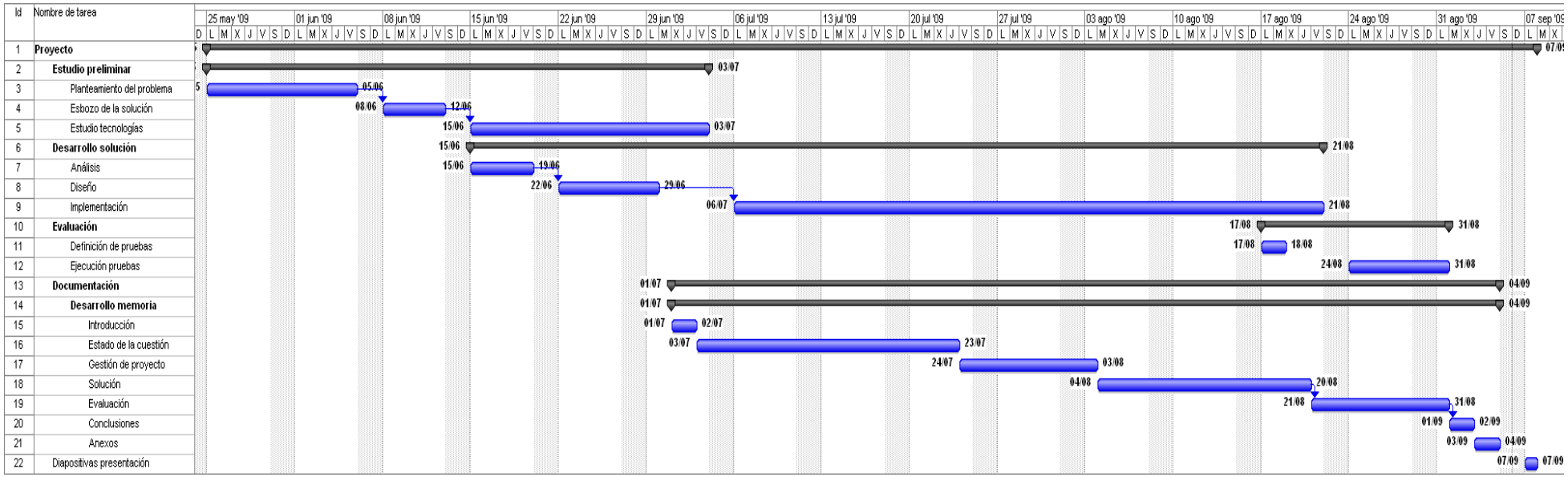

Figura 21: Planificación inicial

Planificación final

Aunque se ha intentado cumplir con la planificación inicial, no siempre ha sido posible, ya que ha habido ciertas tareas que han llevado más tiempo del inicialmente previsto. Tal es el caso del aprendizaje de las tecnologías que se han utilizado en el desarrollo de la aplicación y también del desarrollo del capítulo de la memoria "Planteamiento del problema", el motivo por el que se ha alargado el tiempo previsto para este último se ha debido al esfuerzo de investigación que supone este capítulo, ya que no basta con redactar a partir de conocimientos previos o adquiridos durante el desarrollo de proyecto, sino que hay que buscar mucha información tanto en libros como en Internet lo cual ha llevado más tiempo del previsto. También ha supuesto un esfuerzo mayor del pensado la implementación de la aplicación, ya que el uso de tecnologías desconocidas supone una demora en la resolución de los problemas que surgen. A pesar de estos contratiempos, se ha logrado mantener la fecha de entrega.

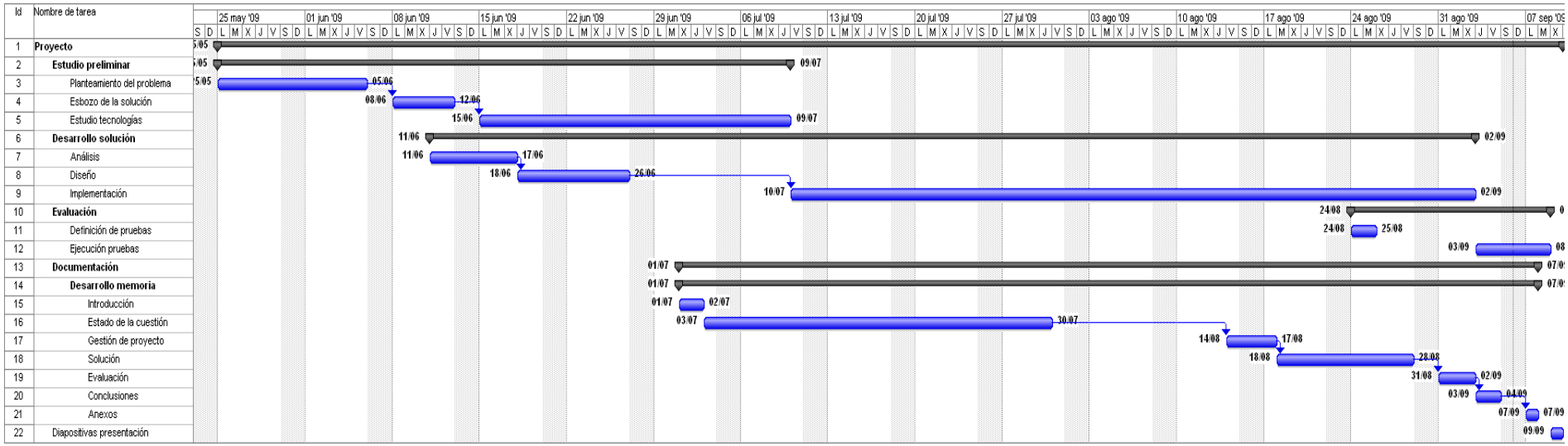

Figura 22: Planificación real

Anexo III: Manual de usuario

Anexo III. Manual de usuario

En esta sección se recoge el manual de usuario que acompaña a la aplicación desarrollada y que sirve como guía en su manejo.

¿Para qué sirve esta aplicación?

Esta aplicación está diseñada con el objetivo de gestionar la base de datos del LEMI (Laboratorio de Estudios Métricos de la Información), facilitando el proceso de inserción y actualización de la información de carácter bibliográfico. Además, permite realizar distintas consultas sobre la base de datos de manera sencilla e intuitiva. Con el fin de poder actualizar la información de la base de datos con independencia del origen de los datos de entrada, la aplicación ofrece un sencillo sistema mediante el cual el usuario puede determinar el formato en que se distribuyen los registros del fichero de entrada.

¿Cómo se accede a la aplicación?

Para acceder a la aplicación, es necesario introducir la URL de la misma en un navegador web (ahora mismo es compatible con los navegadores Internet Explorer y Mozilla Firefox). Una vez introducida la URL se muestra al usuario la página de acceso a la aplicación, donde simplemente es necesario rellenar los campos con el identificador del usuario y su contraseña y pulsar “Aceptar”. Si los datos introducidos son correctos, se muestra la pantalla principal de la aplicación, dando acceso al usuario a todas las operaciones para las que tenga permisos.

¿Cómo se define un nuevo formato de entrada de datos?

La aplicación cuenta con un formato de entrada de datos predefinido, que coincide con el que presentan los registros descargados de la Web of Knowledge. Sin embargo, el usuario puede definir nuevos formatos de entrada para adaptar la herramienta al formato de archivo que puedan presentar otras bases de datos de información bibliográfica. Para ello se deben seguir los siguientes pasos:

1. Escoger del menú de opciones la denominada “Definir formato de entrada”.
2. Escribir un nombre que identifique a este formato de archivo.
3. Definir cómo se identifican los distintos campos del registro.
4. Pulsar “Aceptar”.

De esta forma, se define un nuevo formato de entrada de datos y la próxima vez que se desee insertar nuevos registros a la base de datos, este formato aparecerá en la lista de formatos disponibles.

¿Cómo se actualizan los registros de la base de datos?

Para actualizar la base de datos con nuevos registros o bien modificando los datos de los ya existentes, es necesario seguir los siguientes pasos:

1. Escoger del menú de opciones la identificada como “Actualizar base de datos”.
2. Seleccionar el formato que tiene el archivo con los registros.
3. Escoger el fichero de entrada a partir de la opción
4. Pulsar “Aceptar”.

La aplicación actualiza la base de datos y todas las tablas involucradas en el proceso de manera automática y transparente al usuario. Una vez finalizado este proceso se muestra una pantalla indicando qué número de registros se han insertado nuevos en la base de datos, qué cantidad han sido modificados y cuántos han producido algún tipo de incidencia.

¿Cómo ver qué registros son los que se han insertado nuevos?

Una vez actualizada la base de datos la aplicación muestra unos enlaces con el número de archivos insertados, modificados y que presentan incidencias. Si el usuario pulsa sobre el enlace que indica el número de archivos insertados, la aplicación muestra una tabla con todos los archivos que han sido insertados nuevos.

¿Cómo ver qué registros son los que han sufrido modificaciones?

De la misma forma que los archivos que se han insertado nuevos, sólo que escogiendo el enlace que hace referencia a los archivos modificados.

¿Cómo se realizan consultas sencillas a la base de datos?

Para realizar consultas sencillas sobre una tabla de la base de datos basta con seguir los siguientes pasos:

1. Seleccionar la opción “Consulta con formularios” del menú de opciones.
2. Seleccionar la tabla de la base de datos sobre la que se realizará la consulta.
3. Seleccionar los campos de la tabla que se desean mostrar.
4. Rellenar los campos del formulario según el tipo de consulta a realizar.
5. Pulsar el botón “Aceptar”.

La aplicación muestra en forma de tabla los registros coincidentes con las condiciones de búsqueda impuestas por el usuario.

¿Se pueden realizar consultas más complejas sobre la base de datos?

La aplicación permite realizar consultas de carácter más complejo sobre la base de datos que permitan relacionar varias bases de datos, ordenar los datos de salida, etc. Para ello, el usuario debe tener ciertos conocimientos de SQL para efectuar la consulta mediante sentencias en este lenguaje. Los pasos a seguir son los siguientes:

1. Seleccionar la opción “Consulta SQL” del menú de opciones.
2. Rellenar el cuadro de texto mostrado con la sentencia SQL de consulta.
3. Pulsar el botón “Aceptar”.

La aplicación muestra en forma de tabla los registros coincidentes con la consulta SQL especificada por el usuario. En caso de que la sintaxis de la consulta no sea correcta, se mostrará un mensaje de error.

¿Cómo puedo guardar los resultados que se muestran en las tablas?

Los registros que se muestran en forma de tabla en la aplicación, bien tras un proceso de inserción o la realización de alguna consulta, pueden ser guardados en forma de archivo para su posterior recuperación. Para ello, desde la pantalla donde se muestra la tabla, se pulsa el

botón “Guardar como archivo”, se especifica la ruta y el nombre que se le quiere dar y se pulsa guardar.

¿Existe algún mecanismo de ayuda?

La aplicación cuenta con un sistema de ayuda que recoge la información más importante relacionada con el manejo de la aplicación. Esta distribuida por temas, que coinciden con las operaciones que pueden realizarse con la aplicación y cuya información es la misma que la recogida en este manual de usuario. Para acceder a la ayuda online del sistema hay que seguir estos pasos:

1. Escoger del menú de opciones la denominada “Ayuda”.
2. Seleccionar el tema de ayuda relacionado con la información buscada.

Anexo IV: Prototipo de interfaz

Anexo IV. Prototipo

Para diseñar la interfaz, se presentó un prototipo elaborado con diapositivas en Power Point, para que el usuario lo evaluara y diera su opinión sobre cómo quería que fuera la interfaz de usuario. En esta sección se recogen las distintas imágenes que se presentaron al usuario.

Figura 23: Pantalla de acceso al sistema

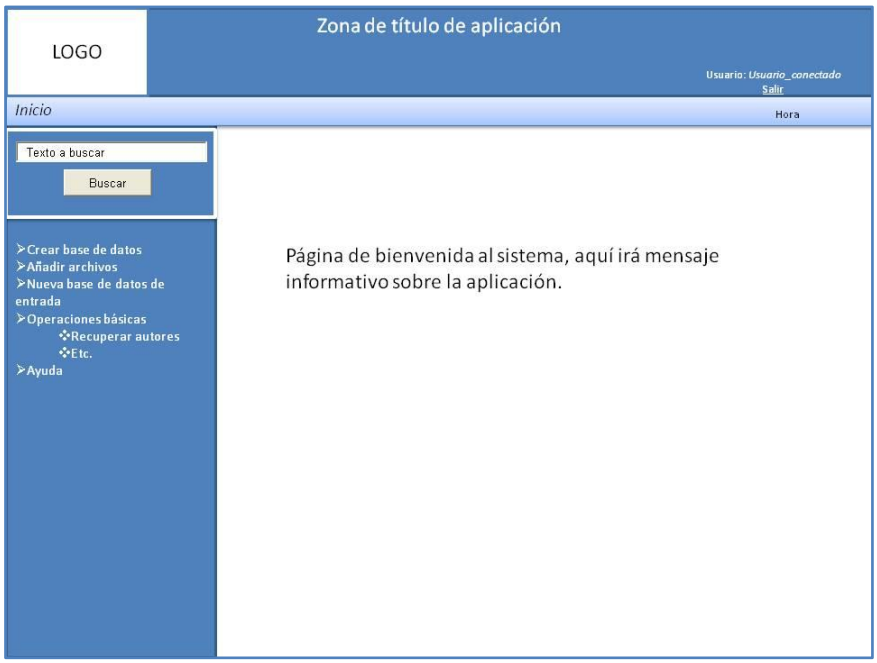

Figura 24: Pantalla de inicio

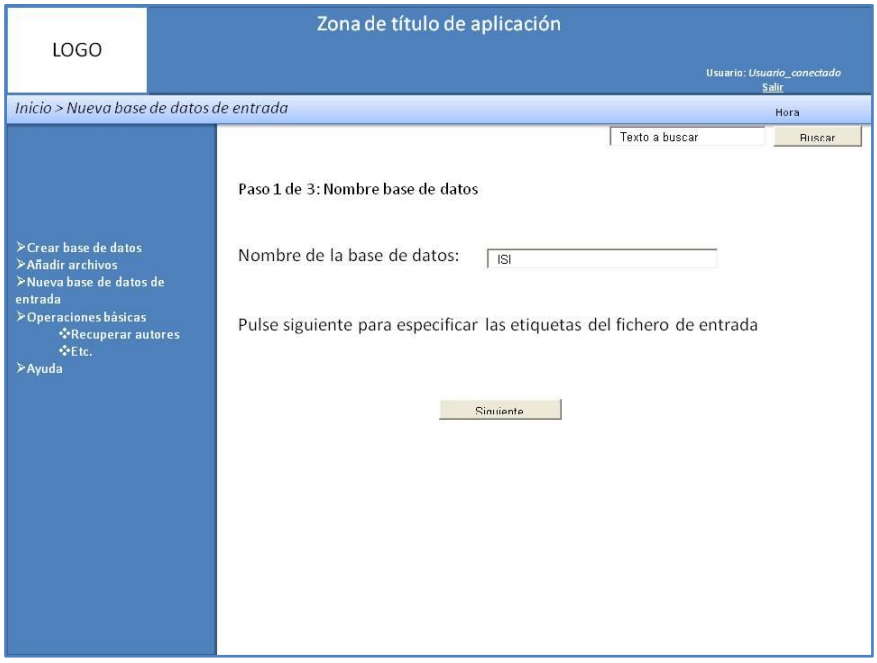

Figura 25: Paso 1 de 3 de la definición de formato de entrada

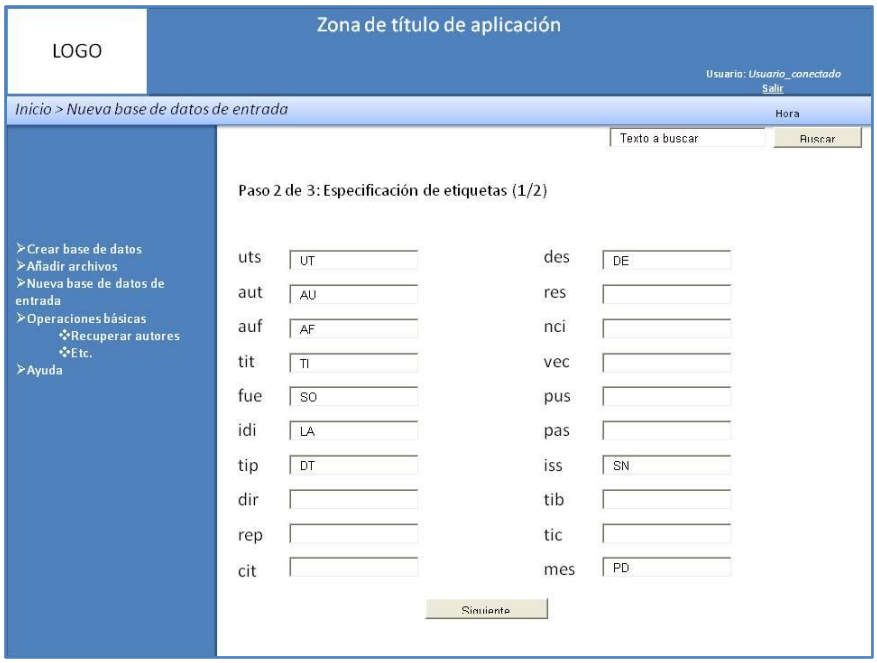

Figura 26: Paso 2 de 3 de la definición de formato de entrada

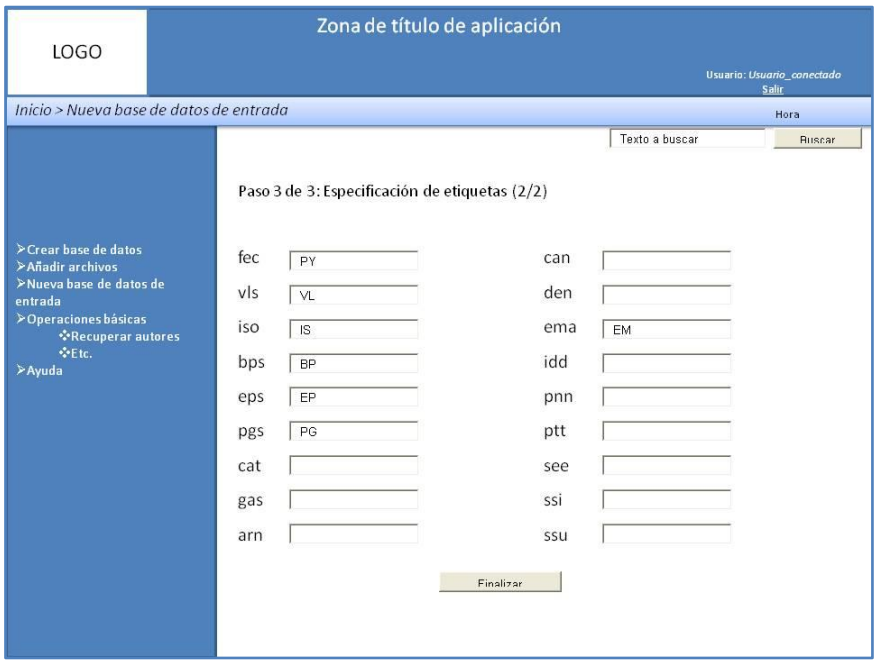

Figura 27: Paso 3 de 3 de la definición de formato de entrada

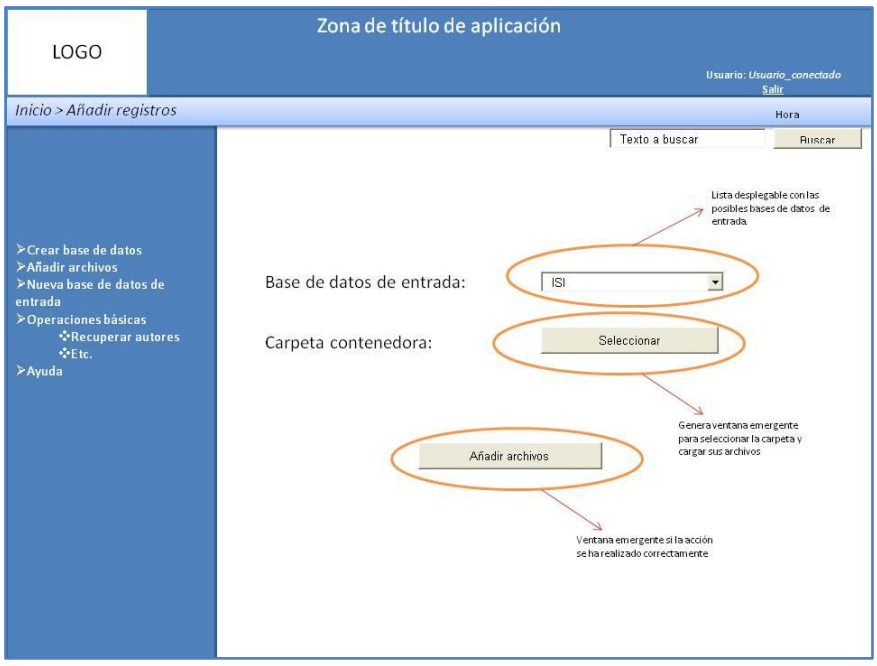

Figura 28: Pantalla de inserción de registros a la base de datos

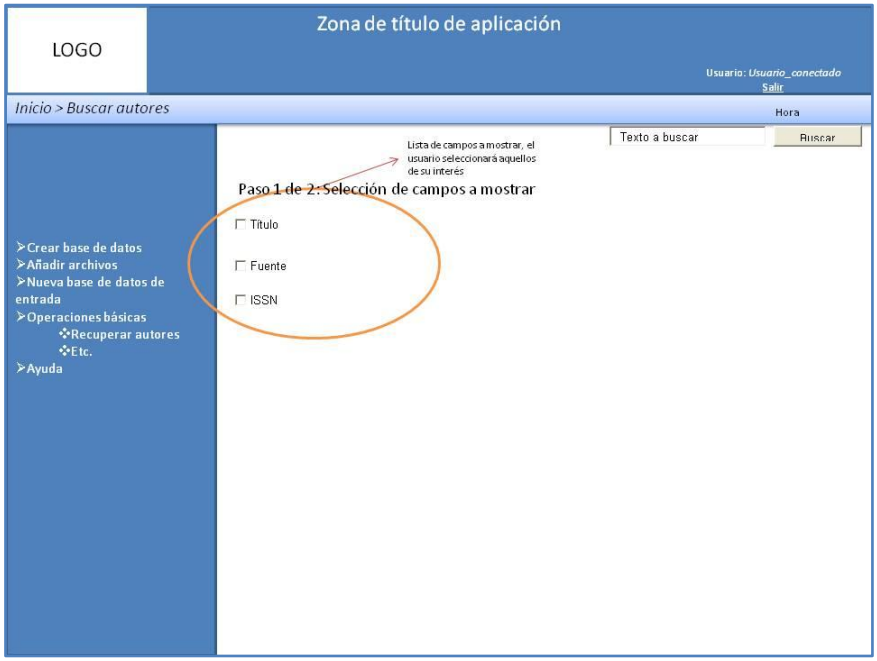

Figura 29: Paso 1 de 2 de la realización de consultas con formularios

Figura 30: Paso 2 de 2 de la realización de consultas con formularios

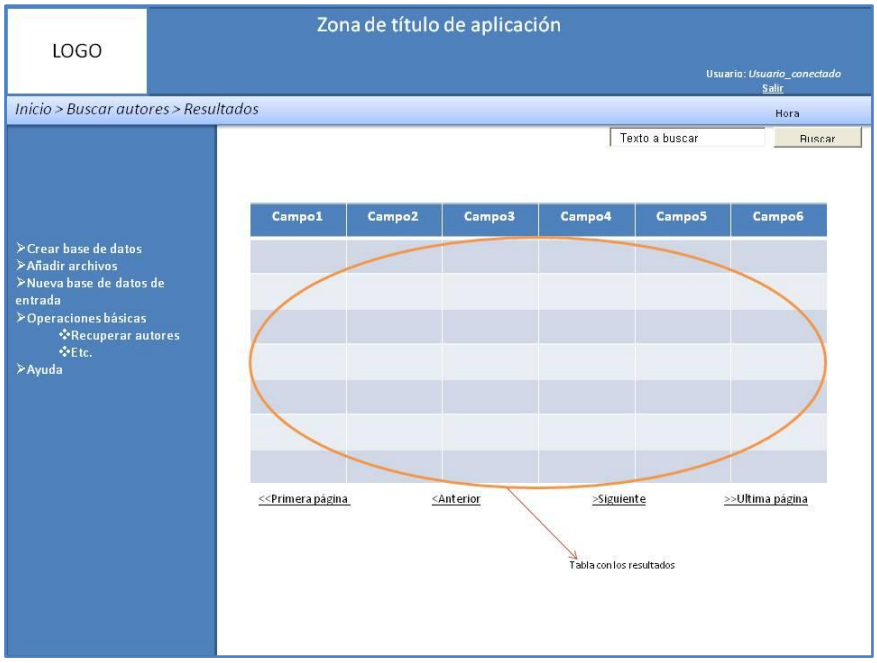

Figura 31: Pantalla que recoge los registros resultado de inserciones o consultas

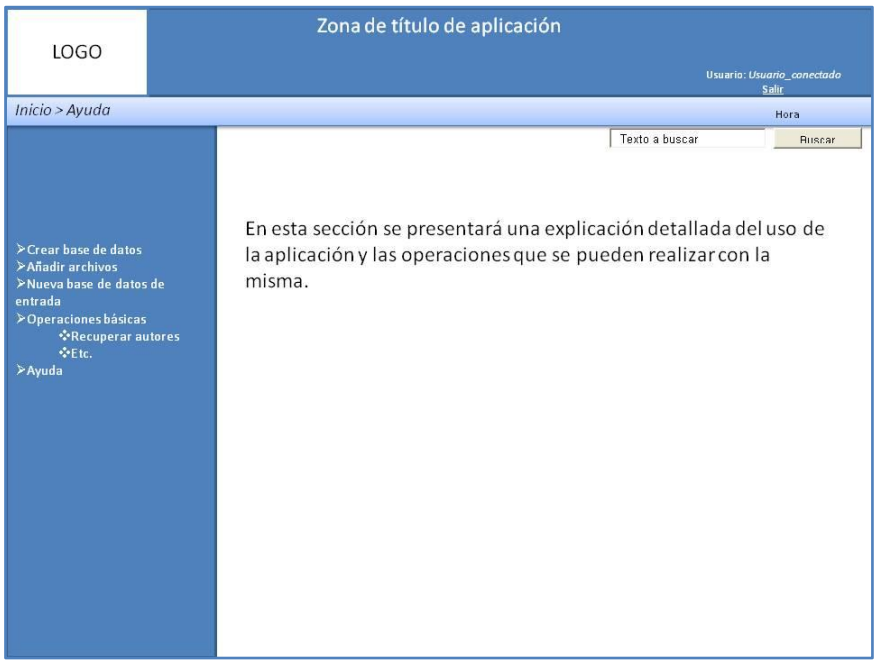

Figura 32: Pantalla de ayuda